UNIVERSIDAD SAN LUIS GONGZA DE ICA

PROGRAMACIÓN ORIENTADA A OBJETOS

para INGENIERÍA MECÁNICA Y ELÉCTICA

2026

INDICE GENERAL (VERSION 12 CAPITULOS)

“Año de la recuperación y consolidación de la economía peruana”

UNIVERSIDAD NACIONAL SAN LUIS GONZAGA

FACULTAD DE INGENIERIA MECANICA ELECTRICA Y ELECTRONICA

TAREA N°1

Terminología orientada a objetos en c++

CURSO: POGRAMACION ORIENTADA A OBJETOS

INTEGRANTES: HERNANDEZ RAMIREZ CRISTOPHER LEONEL

HUAYTA SANTI GEORGE ALEXANDER

PILLPE ROMANI FRANK ROBERT

DOCENTE: ING. ROMAN MUNIVE WILDER ENRIQUE

CICLO: IV

ICA-PERU

2025

Terminología orientada a objetos en c++

I. Introducción a la Orientación a Objetos en C++

1.1 Del Paradigma Procedimental al Orientado a Objetos

La evolución de los lenguajes de programación ha estado marcada por la búsqueda de metodologías que permitan gestionar la creciente complejidad del software. El paradigma procedimental, dominante durante décadas, organiza el software en torno a una secuencia de sentencias y procedimientos o funciones que operan sobre un conjunto de estructuras de datos.[1] En este modelo, los datos y las funciones que los manipulan son entidades separadas. Si bien es efectivo para problemas de escala moderada, este enfoque presenta desafíos a medida que los sistemas crecen, principalmente porque no existe una conexión intrínseca entre los datos y las operaciones que son lógicamente apropiadas para ellos, lo que puede llevar a un acoplamiento no deseado y a una dificultad para mantener la integridad de los datos.[2]

La Programación Orientada a Objetos (POO) representa un cambio fundamental en esta filosofía.[1] En lugar de centrarse en los procedimientos, el paradigma de la POO se organiza en torno al concepto del **objeto**, una entidad que agrupa tanto los datos (atributos) como las operaciones (métodos) que se pueden realizar sobre esos datos.[2] Esta unión de datos y comportamiento en una única unidad cohesiva es la idea central que distingue a la POO.[2] Este cambio no es meramente sintáctico; es una transformación en la forma de descomponer y modelar los problemas. Mientras que la programación procedimental incita a pensar en una secuencia de pasos a seguir, la POO fomenta la identificación de los "actores" o "entidades" del dominio del problema, modelándolos como objetos que encapsulan su propio estado y comportamiento e interactúan entre sí mediante el envío de mensajes.[1] Este enfoque permite un modelado más fiel del mundo real, lo que mejora la comprensión y la relevancia del código, y sienta las bases para la creación de software más modular, reutilizable y mantenible.[4]

1.2 Definición de Clase y Objeto: El Molde y la Instancia

En el núcleo de la POO en C++ se encuentran dos conceptos inseparables: la clase y el objeto.[4]

Una **clase** es un tipo de dato definido por el usuario que actúa como un prototipo, plantilla o molde.[4] Define un conjunto de propiedades (atributos) y comportamientos (métodos) que serán comunes a todos los objetos de ese tipo.[6] Es importante entender que una clase es una abstracción lógica; no ocupa memoria por sí misma (excepto por la información estática), sino que describe cómo se construirán sus instancias.[6] La declaración de una clase, utilizando la palabra clave class, introduce un nuevo tipo en el programa, ampliando el sistema de tipos del lenguaje.[5]

Un **objeto**, por otro lado, es una instancia concreta de una clase.[4] Si una clase es el molde, un objeto es el producto creado a partir de ese molde. La creación de un objeto a partir de una clase se denomina **instanciación**.[4] Cada objeto es una entidad que existe en la memoria y posee su propia copia de los datos (atributos) definidos en la clase, aunque comparte el código de los métodos con otros objetos de la misma clase.[6] La relación entre clase y objeto es análoga a la relación entre un tipo de dato fundamental (como int) y una variable de ese tipo; la clase define el tipo y el objeto es la variable.[5]

1.3 Anatomía de una Clase en C++: Miembros de Datos y Funciones Miembro

Una clase en C++ está compuesta por **miembros**. Estos miembros se dividen en dos categorías principales: miembros de datos y funciones miembro.[5]

Los **miembros de datos** (también conocidos como atributos o variables miembro) son las variables declaradas dentro de la clase. Definen el estado o las características de un objeto.[4] Por ejemplo, una clase Vehiculo podría tener miembros de datos como marca, modelo y velocidad.

Las **funciones miembro** (también conocidas como métodos) son las funciones declaradas dentro de la clase. Definen el comportamiento del objeto y las operaciones que se pueden realizar sobre sus datos.[4] Siguiendo el ejemplo anterior, la clase Vehiculo podría tener funciones miembro como acelerar(), frenar() y obtenerVelocidad(). Una característica fundamental es que las funciones miembro tienen acceso directo a todos los miembros (tanto de datos como otras funciones) de su propia clase, incluyendo aquellos que son

privados.[6]

La sintaxis de una clase en C++ es similar a la de una estructura (struct), pero con una diferencia crucial en el control de acceso por defecto. Por defecto, todos los miembros de una clase definida con la palabra clave class son **privados** (private), lo que significa que no son accesibles desde fuera de la clase. En contraste, los miembros de una struct son **públicos** (public) por defecto.[5] Esta distinción no es arbitraria, sino que refleja una intención de diseño. El uso de struct generalmente indica un agregado simple de datos, donde el acceso directo a sus miembros es aceptable. Por otro lado, el uso de class señala la intención de crear un tipo de objeto más complejo, donde el encapsulamiento y una interfaz pública controlada son los principios rectores.

C++

```
class Vehiculo {
    // Por defecto, estos miembros son privados
    std::string marca;
    int anio;
    int velocidad;

public: // Los miembros a continuación son públicos
    // Declaraciones (prototipos) de las funciones miembro
    void acelerar();
    void frenar();
    int obtenerVelocidad() const; // 'const' indica que este método no modifica el estado del objeto
};
```

1.4 Ejemplo Práctico: La Declaración de una Clase Simple y la Creación de un Objeto

Para consolidar estos conceptos, se presenta un ejemplo completo que declara una clase Registro, define sus miembros y luego instancia objetos a partir de ella.

C++

```cpp
#include <iostream>
#include <string>

// Declaración de la clase 'Registro'
class Registro {
private: // Sección privada (explícita para mayor claridad)
    std::string nombre;
    std::string telefono;

public: // Sección pública: la interfaz de la clase
    // Declaración de los métodos
    void iniciar();
    void entradaDatos();
    void salidaDatos() const;
};

// Implementación de los métodos de la clase 'Registro'
// Se utiliza el operador de resolución de ámbito (::) para indicar que estas funciones pertenecen a la clase 'Registro'
void Registro::iniciar() {
    nombre = "N/A";
    telefono = "000-0000";
    std::cout << "Registro inicializado." << std::endl;
}

void Registro::entradaDatos() {
    std::cout << "Ingrese nombre: ";
    std::getline(std::cin, nombre);
    std::cout << "Ingrese telefono: ";
    std::getline(std::cin, telefono);
}

void Registro::salidaDatos() const {
    std::cout << "--- Datos del Registro ---" << std::endl;
    std::cout << "Nombre: " << nombre << std::endl;
    std::cout << "Telefono: " << telefono << std::endl;
}
```

```
// Instanciación de un objeto global de la clase 'Registro'
Registro registroGlobal;

int main() {
    // Instanciación de un objeto local de la clase 'Registro'
    Registro registroLocal;

    std::cout << "--- Trabajando con registroLocal ---" << std::endl;
    registroLocal.iniciar(); // Llamada a un método usando el operador punto (.)
    registroLocal.entradaDatos();
    registroLocal.salidaDatos();

    std::cout << "\n--- Trabajando con registroGlobal ---" << std::endl;
    registroGlobal.iniciar();
    registroGlobal.salidaDatos();

    return 0;
}
```

En este ejemplo, Registro es la clase. registroGlobal y registroLocal son dos objetos distintos (instancias) de esa clase.[5] Cada uno tiene su propio conjunto de variables nombre y telefono. La llamada a una función miembro, como registroLocal.iniciar(), se realiza utilizando el nombre del objeto seguido del operador de acceso a miembro, el punto (.).[6]

II. El Ciclo de Vida de un Objeto: Constructores y Destructores

Un objeto en C++ tiene un ciclo de vida bien definido: nace (se construye), vive (se utiliza) y muere (se destruye). La gestión de este ciclo es fundamental para garantizar que los objetos se inicialicen correctamente y que los recursos que gestionan se liberen de forma adecuada. C++ proporciona dos tipos de funciones miembro especiales para este propósito: constructores y destructores.

2.1 El Rol del Constructor: Inicialización y Asignación de Recursos

Un **constructor** es una función miembro especial que se ejecuta automáticamente en el momento en que se crea un objeto de una clase.[2] Su propósito principal es inicializar los miembros de datos del objeto y adquirir cualquier recurso necesario (como memoria dinámica o identificadores de archivos), asegurando que el objeto comience su vida en un estado válido y consistente.[2]

Los constructores se identifican porque tienen el mismo nombre que la clase y, a diferencia de otras funciones, no tienen un tipo de retorno, ni siquiera void.[10] Pueden ser sobrecargados, lo que permite diferentes formas de inicializar un objeto.[10]

2.1.1 Constructor por Defecto

Un constructor por defecto es aquel que se puede llamar sin argumentos.[10] Esto incluye tanto a un constructor que no tiene parámetros como a uno cuyos parámetros tienen todos valores por defecto. Si un programador no define ningún constructor para una clase, el compilador de C++ proporcionará un constructor por defecto público e implícito.[10]

C++

```
class Caja {
private:
    double longitud;
    double anchura;
    double altura;
public:
    // Constructor por defecto
    Caja() {
        longitud = 1.0;
        anchura = 1.0;
        altura = 1.0;
        std::cout << "Objeto Caja creado con constructor por defecto." << std::endl;
    }
};
```

```
// Uso:
Caja miCaja; // Llama a Caja::Caja()
```

2.1.2 Constructores Parametrizados

Los constructores parametrizados aceptan uno o más argumentos, lo que permite inicializar un objeto con valores específicos en el momento de su creación.[10] Esta es la forma más común de inicializar objetos, ya que permite que cada instancia tenga un estado inicial único.

C++

```
class Caja {
private:
    double longitud;
    double anchura;
    double altura;
public:
    // Constructor por defecto
    Caja() : longitud(1.0), anchura(1.0), altura(1.0) {} // Usando lista de inicialización de miembros

    // Constructor parametrizado
    Caja(double l, double an, double al) : longitud(l), anchura(an), altura(al) {
        std::cout << "Objeto Caja creado con constructor parametrizado." << std::endl;
    }
};

// Uso:
Caja cajaPequena(10.0, 5.0, 2.0); // Llama a Caja::Caja(double, double, double)
```

El código anterior introduce la **lista de inicialización de miembros** (e.g., : longitud(l)). Esta es la forma preferida de inicializar miembros en C++, ya que es más eficiente que asignar valores dentro del cuerpo del constructor, especialmente para miembros que son objetos de otras clases.

2.1.3 El Constructor de Copia

Un constructor de copia es un tipo especial de constructor parametrizado que se utiliza para inicializar un nuevo objeto como una copia de un objeto existente de la misma clase.[10] Toma como argumento una referencia (generalmente const) a un objeto de su propia clase. El compilador también proporciona un constructor de copia por defecto si no se define uno explícitamente.[10]

C++

```
class Caja {
    //... miembros y otros constructores...
public:
    // Constructor de copia
    Caja(const Caja& otra) : longitud(otra.longitud), anchura(otra.anchura), altura(otra.altura) {
        std::cout << "Objeto Caja creado con constructor de copia." << std::endl;
    }
};

// Uso:
Caja cajaOriginal(10.0, 5.0, 2.0);
Caja cajaCopia(cajaOriginal); // Llama explícitamente al constructor de copia
Caja otraCopia = cajaOriginal; // También llama al constructor de copia (inicialización, no asignación)
```

2.2 El Destructor: Liberación de Recursos y Limpieza

Un **destructor** es la contraparte del constructor. Es una función miembro especial que se invoca automáticamente cuando un objeto está a punto de ser destruido.[2] Esto ocurre cuando un objeto local sale de su ámbito, cuando se usa el operador delete sobre un puntero a un objeto, o cuando el programa termina para objetos globales o estáticos.[13]

El propósito principal del destructor es realizar tareas de limpieza, siendo la más importante la liberación de los recursos que el objeto pudo haber adquirido durante su vida, como la memoria dinámica.[10] El nombre de un destructor es el mismo que el de la clase, pero precedido por una tilde (~). No tiene tipo de retorno y no acepta parámetros, por lo que no puede ser sobrecargado.[10]

C++

```
class VectorDinamico {
private:
    int* datos;
    size_t tamano;
public:
    // Constructor que adquiere memoria
    VectorDinamico(size_t tam) : tamano(tam) {
        datos = new int[tamano]; // Adquisición de recurso (memoria)
        std::cout << "VectorDinamico construido." << std::endl;
    }

    // Destructor que libera la memoria
    ~VectorDinamico() {
        delete datos; // Liberación del recurso
        std::cout << "VectorDinamico destruido." << std::endl;
    }
};

void funcionDePrueba() {
    VectorDinamico v(100); // Se llama al constructor
    //... se usa el vector...
} // El objeto 'v' sale de ámbito aquí, se llama automáticamente al destructor

int main() {
    funcionDePrueba();
    return 0;
}
```

La combinación de constructores que adquieren recursos y destructores que los liberan es la base de una de las técnicas de diseño más potentes y fundamentales de C++: **RAII**

(Resource Acquisition Is Initialization). Este idioma vincula la vida útil de un recurso (como memoria, un archivo o un bloqueo de mutex) a la vida útil de un objeto. Como el destructor tiene la garantía de ser llamado cuando el objeto sale de ámbito, la liberación del recurso se vuelve automática y determinista, previniendo fugas de recursos de manera robusta.[2]

Además, la existencia de constructores de copia y destructores generados por el compilador conduce a un principio de diseño crítico conocido como la **"Regla de Tres/Cinco/Cero"**. Si una clase necesita un destructor definido por el usuario (generalmente porque gestiona un recurso como un puntero a memoria dinámica), es casi seguro que también necesita un constructor de copia y un operador de asignación de copia definidos por el usuario (la "Regla de Tres"). Si no se proporcionan, la copia superficial por defecto que realiza el compilador para el puntero resultará en que dos objetos apunten a la misma memoria.[14] Cuando el primer objeto se destruya, liberará la memoria, dejando al segundo objeto con un puntero colgante. Cuando el segundo objeto se destruya, intentará liberar la misma memoria por segunda vez, lo que provocará un comportamiento indefinido y, a menudo, un fallo del programa.[10]

2.3 El Puntero this: La Autoreferencia Implícita del Objeto

Dentro de cualquier función miembro no estática, C++ proporciona una palabra clave especial: this. El puntero this es un puntero constante que almacena la dirección de memoria del objeto para el cual se ha invocado la función.[15] No se puede declarar ni modificar.[15]

Cuando se accede a un miembro de datos como longitud dentro de un método de la clase Caja, el compilador lo interpreta implícitamente como this->longitud.[17] El puntero this es el mecanismo que permite que una única copia del código de una función miembro opere sobre los datos de diferentes objetos.[17]

Aunque su uso es a menudo implícito, hay situaciones en las que se debe usar explícitamente:

1. **Resolver ambigüedad de nombres:** Cuando un parámetro de una función miembro tiene el mismo nombre que un miembro de datos, this se usa para desambiguar.[18]
   C++
   ```
   class Caja {
   private:
     double longitud;
   public:
     void setLongitud(double longitud) {
   ```

```
        this->longitud = longitud; // 'this->longitud' es el miembro de datos, 'longitud' es el
parámetro
    }
};
```

2. **Devolver el objeto actual:** Para permitir llamadas encadenadas a métodos, una función miembro puede devolver una referencia al objeto actual retornando *this.[16]

```
C++
class Caja {
public:
    Caja& setLongitud(double l) { longitud = l; return *this; }
    Caja& setAnchura(double an) { anchura = an; return *this; }
    Caja& setAltura(double al) { altura = al; return *this; }
private:
    double longitud, anchura, altura;
};

// Uso encadenado:
Caja miCaja;
miCaja.setLongitud(10.0).setAnchura(5.0).setAltura(2.0);
```

3. **Comprobar la auto-asignación:** En la sobrecarga del operador de asignación, this se usa para evitar que un objeto se asigne a sí mismo.[16]

El tipo del puntero this depende de la calificación de la función miembro. En una función no const, su tipo es NombreClase* const. En una función miembro const, su tipo es const NombreClase* const, lo que impide que la función modifique los miembros de datos del objeto.[15]

III. Pilar I: Encapsulamiento y Ocultación de Datos

El encapsulamiento es el primero de los cuatro pilares fundamentales de la Programación Orientada a Objetos. Es el mecanismo que agrupa los datos y los métodos que operan sobre esos datos dentro de una única unidad, la clase, y controla el acceso a su estado interno.[19]

3.1 Principio de Encapsulamiento: Agrupando Datos y

Comportamiento

En su nivel más básico, el encapsulamiento es el proceso de almacenar en una misma sección los elementos que constituyen la estructura (datos) y el comportamiento (métodos) de una abstracción.[21] Este principio es una consecuencia directa de la definición de una clase, que por su naturaleza empaqueta datos y funciones.[5] Sin embargo, el concepto va más allá de la simple agrupación; implica la **ocultación de datos** (data hiding), que consiste en restringir el acceso directo a algunos de los componentes de un objeto.[4]

El objetivo principal de esta ocultación es proteger la integridad de la información del objeto.[20] Al evitar que el código externo modifique directamente el estado interno de un objeto, se previene la corrupción accidental de datos y se garantiza que el objeto se mantenga siempre en un estado coherente y válido.[4]

3.2 Control de Acceso en C++: Los Especificadores public, private y protected

C++ implementa el encapsulamiento y la ocultación de datos a través de tres palabras clave conocidas como **especificadores de acceso**. Estos especificadores definen el nivel de visibilidad de los miembros de una clase.[23]

- **public**: Los miembros declarados como public son accesibles desde cualquier parte del programa, tanto desde dentro de la propia clase como desde código externo que utilice un objeto de esa clase.[23] Los miembros públicos constituyen la **interfaz** de la clase, es decir, la forma en que los usuarios interactúan con el objeto.
- **private**: Los miembros declarados como private solo pueden ser accedidos por las funciones miembro y las funciones o clases friend de la misma clase.[24] Son completamente inaccesibles desde el exterior. Este es el nivel de protección más alto y es el especificador por defecto para los miembros de una class.[6] Los miembros privados forman la **implementación** interna del objeto.
- **protected**: Los miembros declarados como protected son similares a los private en el sentido de que no son accesibles desde el código externo. Sin embargo, sí pueden ser accedidos por las funciones miembro de la misma clase y, crucialmente, por las funciones miembro de las **clases derivadas** (clases hijas).[23] Este especificador está diseñado específicamente para su uso en el contexto de la herencia.

C++

```
class Empleado {
public: // Interfaz pública
    Empleado(const std::string& n, double s) : nombre(n), salario(s) {}

    void mostrarNombre() const {
        std::cout << "Nombre: " << nombre << std::endl;
    }

    double obtenerSalario() const { // 'Getter' para acceso controlado
        return salario;
    }

private: // Implementación privada
    std::string nombre;
    double salario;

protected: // Accesible para clases derivadas
    int idEmpleado;
};
```

En este ejemplo, nombre y salario son privados, protegiendo su integridad. El acceso al salario desde el exterior solo es posible a través del método público obtenerSalario(), que permite la lectura pero no la modificación directa.

3.3 La Interfaz Pública vs. la Implementación Privada

La separación que crean los especificadores de acceso entre la interfaz pública y la implementación privada es uno de los beneficios más significativos del encapsulamiento, ya que fomenta la **mantenibilidad** del código.[27] La interfaz pública de una clase es un contrato con sus usuarios. Mientras este contrato (los prototipos de las funciones públicas) se mantenga estable, el autor de la clase tiene la libertad de cambiar completamente la implementación interna sin afectar al código que utiliza la clase.[20]

Por ejemplo, una clase que almacena una colección de elementos podría usar un std::vector internamente. Si más adelante se determina que un std::list sería más eficiente, se puede cambiar la implementación privada. Mientras los métodos públicos como agregarElemento() y obtenerElemento() sigan funcionando de la misma manera, el código cliente no necesitará ninguna modificación. Esta separación reduce el acoplamiento entre las diferentes partes de un sistema y facilita la evolución y el mantenimiento del software a largo plazo.[4]

Aunque protected es esencial para la herencia, representa una forma más débil de encapsulamiento que private. Un miembro private solo es visible dentro de su propia clase, limitando el impacto de cualquier cambio a ese único ámbito. En cambio, un miembro protected es visible para toda una jerarquía de clases derivadas.[26] Un cambio en un miembro protected puede requerir modificaciones en todas las clases hijas que dependen de él, haciendo que la jerarquía sea más frágil. Por esta razón, una buena práctica de diseño es mantener los datos como private siempre que sea posible y exponer la funcionalidad a las clases derivadas a través de métodos protected o public, en lugar de exponer los datos directamente.

3.4 Funciones y Clases friend: Excepciones Controladas al Encapsulamiento

En ocasiones, puede ser necesario que una función externa o una clase distinta necesite un acceso privilegiado a los miembros private y protected de otra clase. C++ proporciona un mecanismo para esto a través de la palabra clave friend.[28]

Una **función friend** es una función que, aunque no es miembro de una clase, se le concede permiso para acceder a sus miembros privados y protegidos.[28] Se declara dentro de la clase a la que accederá, precedida por la palabra clave friend.

Una **clase friend** es una clase a la que se le otorga acceso total a los miembros privados y protegidos de otra clase. Esto significa que todos los métodos de la clase amiga pueden acceder a los miembros no públicos de la clase que la declaró su amiga.[28]

C++

```
class Rectangulo; // Declaración anticipada
```

```
class CostoPintura {
public:
    double obtenerCosto(const Rectangulo& rect);
};

class Rectangulo {
private:
    double anchura;
    double altura;
public:
    Rectangulo(double an, double al) : anchura(an), altura(al) {}

    // La función obtenerCosto de la clase CostoPintura es amiga de Rectangulo
    friend double CostoPintura::obtenerCosto(const Rectangulo& rect);
};

// Implementación de la función amiga
double CostoPintura::obtenerCosto(const Rectangulo& rect) {
    // Puede acceder a los miembros privados de Rectangulo
    double area = rect.anchura * rect.altura;
    return area * 5.0; // Suponiendo un costo de 5.0 por unidad de área
}
```

El mecanismo friend rompe intencionadamente el encapsulamiento y debe usarse con moderación.[31] Es apropiado en situaciones donde dos o más clases están tan estrechamente interrelacionadas que conceptualmente forman parte de la misma interfaz, como puede ser el caso de una clase Matriz y una clase Vector, o en la sobrecarga de ciertos operadores como << y >> para flujos de E/S.[29]

IV. Pilar II: Abstracción y Simplificación de la Complejidad

La abstracción es el segundo pilar de la POO y se refiere al proceso de ocultar los detalles complejos de la implementación y exponer solo las funcionalidades esenciales de un objeto.[4] Es un principio de diseño que se centra en la simplificación.

4.1 El Concepto de Abstracción en el Diseño de Software

La abstracción consiste en identificar las características y comportamientos esenciales de una entidad y representarlos en un modelo simplificado, ignorando los detalles irrelevantes o secundarios.[19] El objetivo es reducir la complejidad y permitir que los desarrolladores interactúen con un concepto a un nivel superior, sin necesidad de preocuparse por cómo funciona internamente.[4]

Un ejemplo clásico es la conducción de un automóvil: para usarlo, un conductor solo necesita conocer la interfaz (volante, pedales, palanca de cambios). No es necesario entender la mecánica interna del motor de combustión, la transmisión o el sistema eléctrico.[32] De manera similar, en software, la abstracción permite a los programadores utilizar componentes complejos a través de interfaces simples y bien definidas.

4.2 Implementación de la Abstracción: Clases Abstractas

En C++, una de las herramientas primarias para implementar la abstracción es la **clase abstracta**. Una clase abstracta es una clase que no puede ser instanciada directamente; es decir, no se pueden crear objetos de su tipo.[19] Está diseñada específicamente para servir como una clase base en una jerarquía de herencia, definiendo una interfaz común para todas sus clases derivadas.[34]

Una clase se convierte en abstracta si contiene al menos una **función virtual pura**.[34] Aunque no se pueden crear objetos de una clase abstracta, sí es posible declarar punteros y referencias a ella, lo cual es fundamental para el polimorfismo.[34]

La incapacidad de instanciar una clase abstracta no es una limitación, sino una característica de diseño poderosa. Actúa como un mecanismo de seguridad en tiempo de compilación que garantiza que solo se puedan crear objetos de clases concretas que han implementado completamente la interfaz requerida. El compilador fuerza a las clases derivadas a cumplir con el "contrato" establecido por la clase base abstracta, impidiendo la creación de objetos "incompletos" y trasladando la detección de errores del tiempo de ejecución al tiempo de compilación.[34]

4.3 Funciones Virtuales Puras: Definiendo Contratos de Interfaz

Una **función virtual pura** es una función virtual que se declara en una clase base pero no tiene una implementación en esa clase. Su declaración se distingue por el especificador = 0 al final.[34]

Sintaxis:
virtual tipo nombre_funcion(parametros) = 0;
La presencia de una función virtual pura en una clase la convierte automáticamente en una clase abstracta.[34] Más importante aún, una función virtual pura establece un contrato obligatorio para todas las clases derivadas concretas: deben proporcionar una implementación (sobrescribir) para esa función.[34] Si una clase derivada no sobrescribe todas las funciones virtuales puras que hereda, esa clase derivada también se convierte en abstracta.[34]

C++

```
// 'Figura' es una clase abstracta porque contiene una función virtual pura.
class Figura {
public:
    // Función virtual pura: define un contrato.
    // Toda clase derivada concreta DEBE implementar area().
    virtual double area() const = 0;

    // Un destructor virtual es esencial en clases base polimórficas.
    virtual ~Figura() {}
};

class Circulo : public Figura {
private:
    double radio;
public:
    Circulo(double r) : radio(r) {}

    // Implementación obligatoria del contrato.
    double area() const override {
        return 3.14159 * radio * radio;
```

```
    }
};

class Rectangulo : public Figura {
private:
    double anchura, altura;
public:
    Rectangulo(double an, double al) : anchura(an), altura(al) {}

    // Implementación obligatoria del contrato.
    double area() const override {
        return anchura * altura;
    }
};

int main() {
    // Figura fig; // Error de compilación: no se puede instanciar una clase abstracta.
    Figura* pFig1 = new Circulo(10.0);
    Figura* pFig2 = new Rectangulo(5.0, 4.0);

    std::cout << "Area del circulo: " << pFig1->area() << std::endl;
    std::cout << "Area del rectangulo: " << pFig2->area() << std::endl;

    delete pFig1;
    delete pFig2;
    return 0;
}
```

C++ no tiene una palabra clave interface como otros lenguajes (por ejemplo, Java o C#). En su lugar, el concepto de interfaz se simula idiomáticamente mediante clases abstractas que contienen únicamente funciones virtuales puras (y un destructor virtual).[32] Una clase de este tipo define un contrato puro sin proporcionar ninguna implementación, obligando a las clases que la heredan a implementar toda la interfaz.[19]

4.4 Diferencia entre Abstracción y Encapsulamiento

Aunque están relacionados y a menudo se confunden, abstracción y encapsulamiento son conceptos distintos.[4]

- **Encapsulamiento** es un mecanismo de **implementación**. Se ocupa de agrupar datos y métodos en una clase y de controlar el acceso a ellos (usando public, private, protected). Su objetivo es ocultar los datos y proteger la integridad del objeto.
- **Abstracción** es un principio de **diseño**. Se ocupa de ocultar la complejidad y de exponer solo la funcionalidad esencial a través de una interfaz simplificada. Su objetivo es manejar la complejidad del sistema.

En resumen, el encapsulamiento es una de las técnicas que se pueden utilizar para lograr la abstracción. Se ocultan los detalles internos de un objeto (encapsulamiento) para presentar una vista simplificada de lo que el objeto hace (abstracción).[4]

REFERENCIAS

1. PROGRAMACIÓN ORIENTADA A OBJETOS CON C++ ¡o cómo aprender C++ en 1 hora! - elhacker.INFO, fecha de acceso: octubre 11, 2025, https://elhacker.info/manuales/Lenguajes%20de%20Programacion/C++/C++%20Programaci%C3%B3n%20OOP%20con%20C++.pdf
2. programación orientada a objetos (poo) con c++, fecha de acceso: octubre 11, 2025, http://www.lcc.uma.es/~eat/services/poo_c/fra2.html
3. Los Cuatros Pilares de La Programación Orientada A Objetos. | PDF - Scribd, fecha de acceso: octubre 11, 2025, https://fr.scribd.com/document/720067243/Los-cuatros-pilares-de-la-programacion-orientada-a-objetos
4. 4 pilares de la Programación Orientada a Objetos - CEV, fecha de acceso: octubre 11, 2025, https://www.cev.com/blog/4-pilares-de-la-programacion-orientada-a-objetos
5. 2 - Clases y objetos en C++ - GRC Informatica, fecha de acceso: octubre 11, 2025, http://www.grch.com.ar/docs/unlu.poo/Clases-objetos%20C++.pdf
6. Programación en C++/Objetos y Clases - Wikilibros, fecha de acceso: octubre 11, 2025, https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B/Objetos_y_Clases
7. C++ POO (Programación Orientada a Objetos) - Cimat, fecha de acceso: octubre 11, 2025, https://www.cimat.mx/~pepe/cursos/lenguaje_2012/slides/slide_40.pdf
8. Access specifiers - cppreference.com - C++ Reference, fecha de acceso: octubre 11, 2025, http://en.cppreference.com/w/cpp/language/access.html
9. Tema 4 Clases y objetos en C++, fecha de acceso: octubre 11, 2025,

http://www.lcc.uma.es/~lopez/lp2/apuntes/04-objetos/objetos.pdf
10. Constructor and Destructor in C++ - Scaler Topics, fecha de acceso: octubre 11, 2025, https://www.scaler.com/topics/cpp/constructor-and-destructor-in-cpp/

Informe de Investigación: Punteros en C++ en el Entorno Visual Studio 2022

Resumen

Los punteros son una de las características más poderosas y fundamentales del lenguaje C++. Permiten acceder y manipular directamente la memoria del sistema, lo que brinda gran flexibilidad y eficiencia. Este informe analiza el concepto, uso y buenas prácticas de los punteros en C++ moderno, ilustrando su implementación en el entorno Visual Studio 2022. Se incluyen ejemplos prácticos, referencias académicas y recursos de aprendizaje en español.

Palabras clave — C++, punteros, memoria, dirección, desreferencia, Visual Studio 2022, programación orientada a objetos.

I. INTRODUCCIÓN

En C++, un puntero es una variable que almacena la dirección de memoria de otra variable [1]. A diferencia de las variables tradicionales que contienen datos, los punteros contienen referencias a ubicaciones de memoria, permitiendo acceder y modificar datos de manera indirecta.

El conocimiento de punteros es esencial para comprender estructuras de datos dinámicas (listas enlazadas, árboles, etc.) y conceptos avanzados como la Programación Orientada a Objetos (POO), ya que los objetos son frecuentemente manipulados mediante punteros y referencias [2].

II. CONCEPTOS FUNDAMENTALES DE LOS PUNTEROS

A. Definición

Un puntero se declara especificando el tipo de dato al que apuntará, seguido de un asterisco *.

Ejemplo:

```
int *ptr;
```

Aquí, ptr es un puntero que puede almacenar la dirección de memoria de una variable entera [3].

B. Operador de Dirección (&)

El operador & obtiene la dirección de memoria de una variable.

Ejemplo:

```
int num = 20;

ptr = &num;
```

En este caso, ptr almacena la dirección donde está guardado num.

C. Operador de Desreferencia (*)

El operador * accede al valor almacenado en la dirección apuntada.

cout << *ptr; // Imprime 20

El operador de desreferencia convierte el puntero nuevamente en el valor original [4].

D. Tipos de Punteros

El tipo del puntero debe coincidir con el tipo de la variable apuntada. Por ejemplo, un puntero int* no puede almacenar la dirección de un char. Esto garantiza seguridad en el manejo de memoria [5].

III. EJEMPLO PRÁCTICO EN VISUAL STUDIO C++ 2022

El siguiente código puede ser implementado y ejecutado directamente en **Visual Studio 2022**:

```
#include <iostream>
using namespace std;

int main() {
    int num = 20;          // Variable normal
    int* dirNum;           // Declaración del puntero
    dirNum = &num;         // Asignar dirección de memoria

    cout << "Valor de num: " << num << endl;
    cout << "Direccion de num: " << &num << endl;
    cout << "Valor almacenado en el puntero: " << dirNum << endl;
    cout << "Valor apuntado por el puntero: " << *dirNum << endl;

    return 0;
}
```

Salida esperada:

```
Valor de num: 20
Direccion de num: 0x7ffd2b0a0c4
Valor almacenado en el puntero: 0x7ffd2b0a0c4
Valor apuntado por el puntero: 20
```

IV. APLICACIONES DE LOS PUNTEROS EN LA PROGRAMACIÓN ORIENTADA A OBJETOS

En la POO, los punteros se usan comúnmente para:

- Manipular **objetos dinámicos** creados con `new`.
- Implementar **herencia y polimorfismo** mediante punteros a clases base.
- Controlar recursos del sistema (memoria, archivos, hardware).

Ejemplo:

```
class Persona {
public:
    string nombre;
    Persona(string n) : nombre(n) {}
    void mostrar() { cout << "Nombre: " << nombre << endl; }
};

int main() {
    Persona* p1 = new Persona("Carlos");
    p1->mostrar();
    delete p1;  // Liberar memoria
    return 0;
}
```

V. CONCLUSIONES

Los punteros son herramientas críticas en C++ que proporcionan control directo sobre la memoria y son la base de estructuras de datos dinámicas y de la programación orientada a objetos. Visual Studio 2022 ofrece un entorno poderoso para practicar su uso, con depuración visual de direcciones y valores en tiempo real.

Referencias

[1] B. Stroustrup, Programming: Principles and Practice Using C++, Addison-Wesley, 2014.

[2] H. Schildt, C++: The Complete Reference, McGraw-Hill Education, 2018.

[3] Universidad Politécnica de Madrid, “Apuntes de Punteros en C++”, 2020.

[4] TutorialesProgramacionYa, “Curso de C++: Punteros”, disponible en: https://www.tutorialesprogramacionya.com.

[5] Microsoft Learn, “Punteros y direcciones de memoria en C++”, disponible en: https://learn.microsoft.com/es-es/cpp

.

Guía de Instalación de Visual Studio 2022 Community para C++

Pillpe Romaní Frank Robert

Universidad San Luis Gonzaga

***Resumen*— El presente informe tiene como objetivo guiar paso a paso la instalación y configuración del entorno de desarrollo Visual Studio 2022 Community para la programación en C++. Esta herramienta es esencial para los cursos de Programación Orientada a Objetos, Algoritmos y Simulación Computacional dentro de la carrera de Ingeniería Mecánica Eléctrica.**

Visual Studio 2022 es un entorno de desarrollo integrado (IDE) de Microsoft que permite compilar, depurar y ejecutar proyectos en múltiples lenguajes, entre ellos C++. En este documento se describe el proceso de descarga, instalación, selección de cargas de trabajo y verificación del entorno.

Palabras clave — Visual Studio 2022, C++, IDE, compilador, instalación.

Abstract- This report aims to provide a step-by-step guide through the installation and configuration of the Visual Studio 2022 Community development environment for C++ programming. This tool is essential for Object-Oriented Programming, Algorithms, and Computational Simulation courses within the Mechanical and Electrical Engineering program.

Visual Studio 2022 is a Microsoft integrated development environment (IDE) that allows you to compile, debug, and run projects in multiple languages, including C++. This document describes the process of downloading, installing, selecting workloads, and verifying the environment.

Keywords — Visual Studio 2022, C++, IDE, compiler, installation.

I. INTRODUCIÓN

Visual Studio es un entorno de desarrollo integrado (IDE) ampliamente utilizado por programadores, estudiantes e ingenieros para desarrollar aplicaciones en diversos lenguajes como C++, C#, Python y JavaScript.

La versión Community 2022 es gratuita y especialmente recomendada para uso educativo y proyectos personales, ofreciendo todas las herramientas necesarias para compilar, depurar y ejecutar programas de consola, escritorio o sistemas más complejos.

II. REQUISITOS

Antes de instalar Visual Studio, asegúrate de cumplir los siguientes requisitos:

Antes de instalar Visual Studio 2022 Community, es fundamental conocer las condiciones mínimas que debe cumplir el sistema.

Un equipo que no cumpla estos parámetros puede presentar errores, lentitud o fallas de compilación.

Requisitos mínimos del sistema:

- Sistema operativo: Windows 10 o Windows 11 (solo versiones de 64 bits).
- Las versiones de 32 bits no son compatibles debido a las exigencias de memoria del IDE y sus componentes.

- Procesador: Intel Core i5 o AMD Ryzen 5.
- Se recomienda una velocidad base de 2.0 GHz o superior, ya que el compilador de C++ requiere procesamiento constante durante la construcción de proyectos grandes.

- Memoria RAM: Mínimo 8 GB.
- El entorno y las librerías de C++ utilizan memoria dinámica para compilar, ejecutar y depurar proyectos simultáneamente.

- Espacio en disco: 10 GB libres.
- Durante la instalación, el IDE descarga librerías, SDKs y herramientas adicionales; el espacio debe estar disponible en la unidad principal (normalmente C:).

- Conexión a Internet: Obligatoria.
- Visual Studio utiliza un instalador en línea que descarga los componentes a medida que se seleccionan las cargas de trabajo.

- Cuenta Microsoft: Necesaria.
- Se requiere una cuenta gratuita de Microsoft para activar la licencia Community y sincronizar las configuraciones del entorno

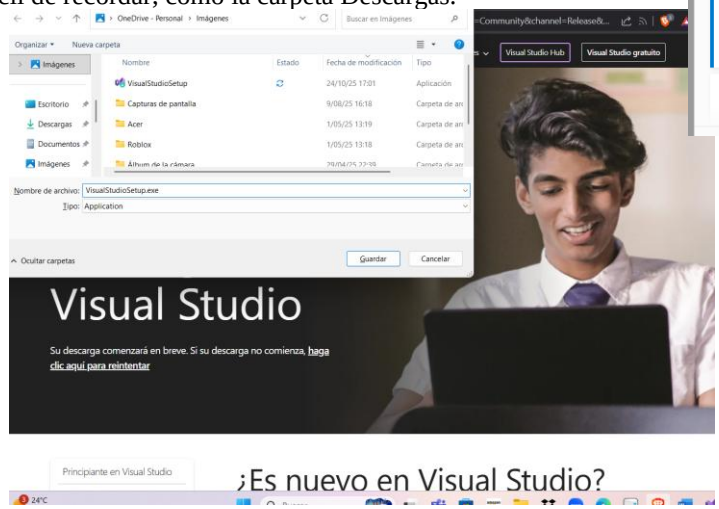
III. DESCARGA

Ingresa al sitio official:
https://visualstudio.microsoft.com/es/vs/community/

En la sección de descargas, selecciona “Visual Studio 2022 Community”.

Haz clic en “Descargar Visual Studio 2022”.

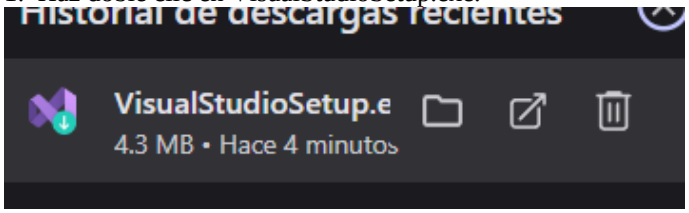
Guarda el archivo VisualStudioSetup.exe en una ubicación fácil de recordar, como la carpeta Descargas.



IV. INSTALACIÓN

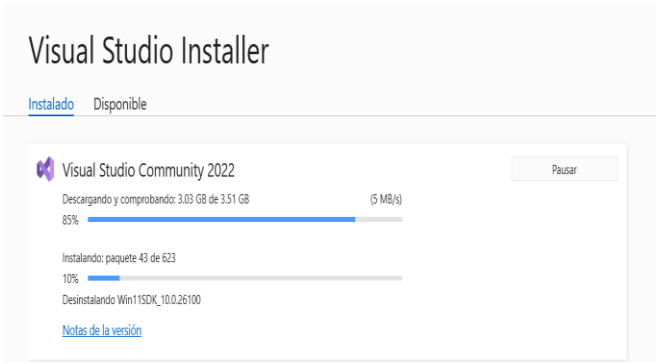
Iniciar instalador

1. Haz doble clic en VisualStudioSetup.exe.



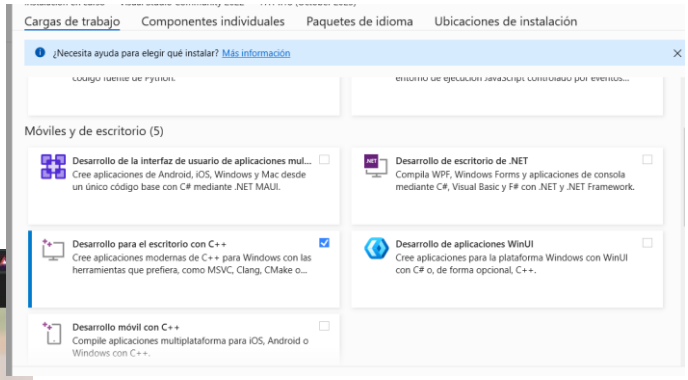
2. Espera a que el instalador descargue los componentes iniciales.

3. Cuando aparezca la ventana principal, continúa con la selección de cargas de trabajo.



V. SELECCIÓN DE COMPONENTES

1. En la pestaña “Cargas de trabajo”, marca “Desarrollo de escritorio con C++”.

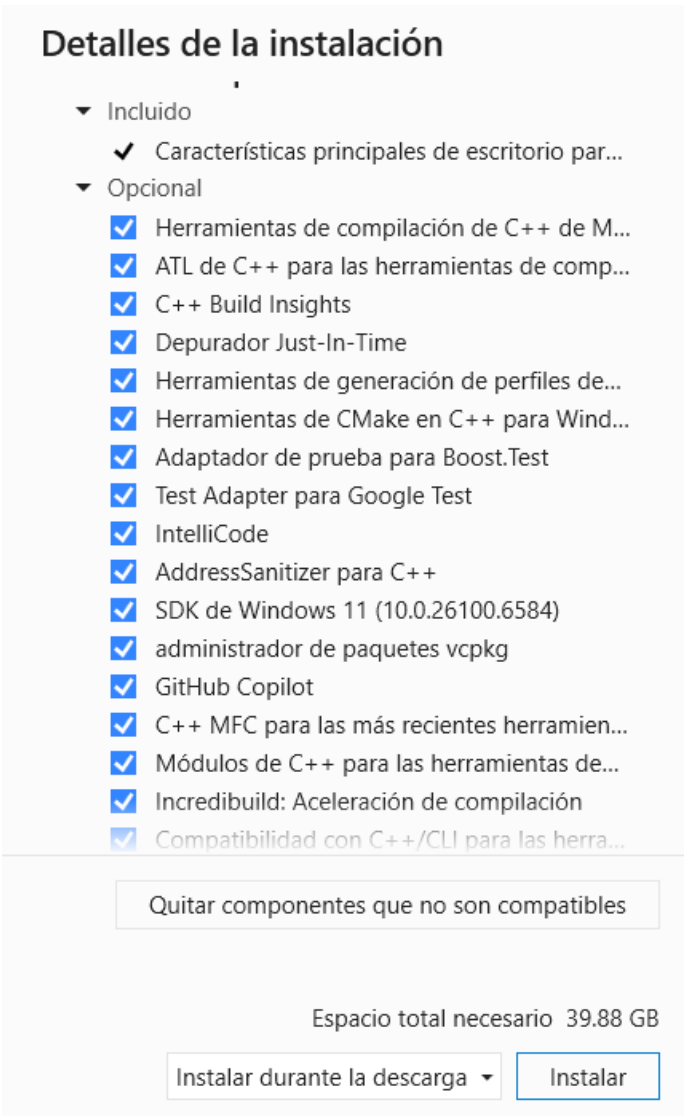


2. Verifica que estén activados:

3. Compilador MSVC.

4. Windows SDK

5. Herramientas de depuración y CMake.



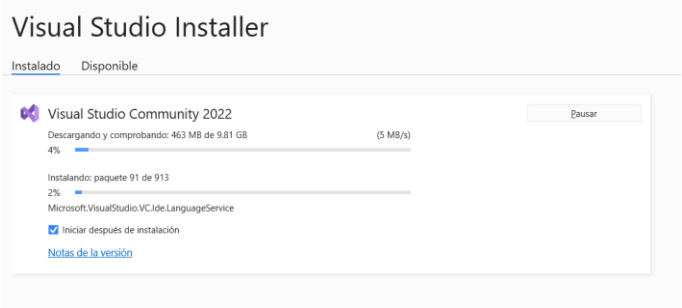
6. Haz clic en “Instalar” para comenzar.

VI. PROCESO DE INSTALACIÓN

1. La descarga e instalación pueden tomar entre 20 y 45 minutos.

2. No apagues el equipo ni cierres la ventana del instalador.

3. Al finalizar, haz clic en “Iniciar Visual Studio”.



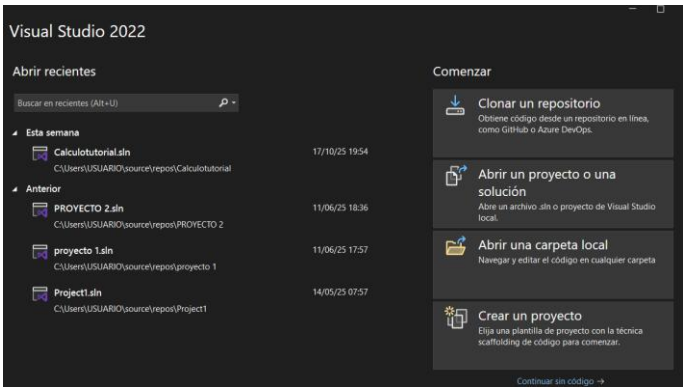
VII. CONFIGURACIÓN

1. Inicia sesión con tu cuenta Microsoft (puedes crear una gratuitamente si no tienes).

2. Elige un tema visual: se recomienda el modo oscuro por comodidad y menor fatiga visual.

3. Selecciona el entorno predeterminado de C++.

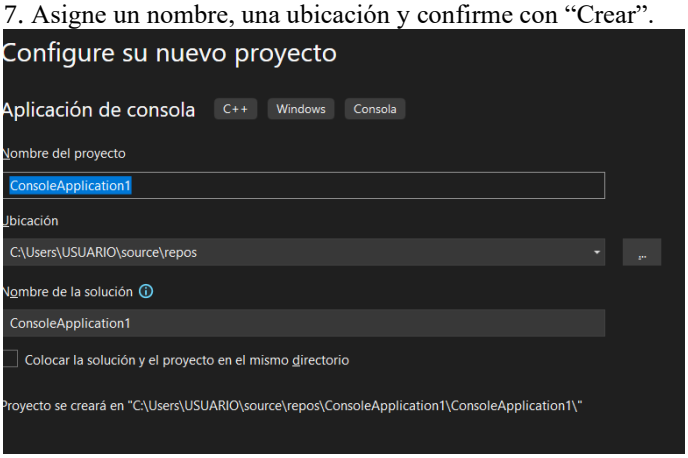
4. Guarda la configuración y accede al entorno principal.



5. Escoger donde dice crear un-Proyecto

6. Presionar donde dice aplicacion de consola

7. Asigne un nombre, una ubicación y confirme con “Crear”.

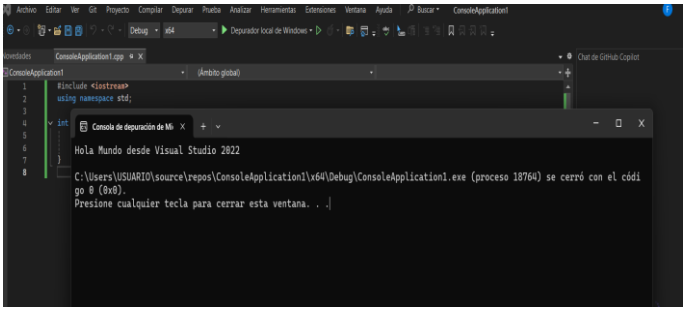


VIII. CÓDIGO DE PRUEBA

En el archivo main.cpp, escribe el siguiente código:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hola Mundo desde Visual Studio 2022" << endl;
    return 0;
}
```

Presiona Ctrl + F5 para ejecutar el programa.
Si la consola muestra el texto correctamente, el entorno está funcionando.



IX. CONCLUSIONES

1. Visual Studio 2022 Community es una plataforma estable y gratuita que facilita el desarrollo en C++.

2. Cumplir los requisitos mínimos garantiza un proceso de instalación sin errores.

3. La correcta selección de la carga de trabajo “Desarrollo de escritorio con C++” es clave para disponer del compilador y librerías necesarias.

4. Verificar la ejecución de un proyecto de prueba permite asegurar que el entorno funciona correctamente.

5. Este IDE fortalece las competencias prácticas de los estudiantes en programación, simulación y automatización, áreas esenciales de la ingeniería moderna.

REFERENCIAS

[1] Microsoft, *Visual Studio 2022 Community Documentation*, https://visualstudio.microsoft.com/es/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&passive=false&cid=2030

[2] RETRO, «YOUYUBE,» RETRO, 1 JUNIO 2023. [En línea]. Available: https://visualstudio.microsoft.com/es/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&passive=false&cid=2030.

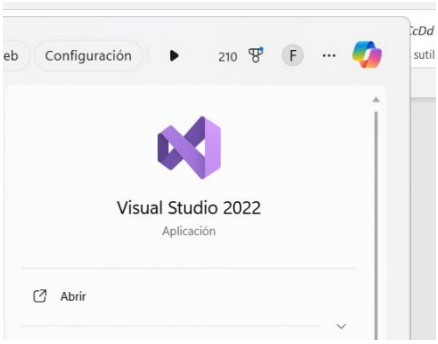
CALCULA LA POTENCIA ELÉCTRICA DE UN MOTOR, USANDO LA FÓRMULA BÁSICA

Aplicación de escritorio en C# (Windows Forms) que permite ingresar el voltaje y la corriente de un motor y calcula su potencia eléctrica mediante la fórmula

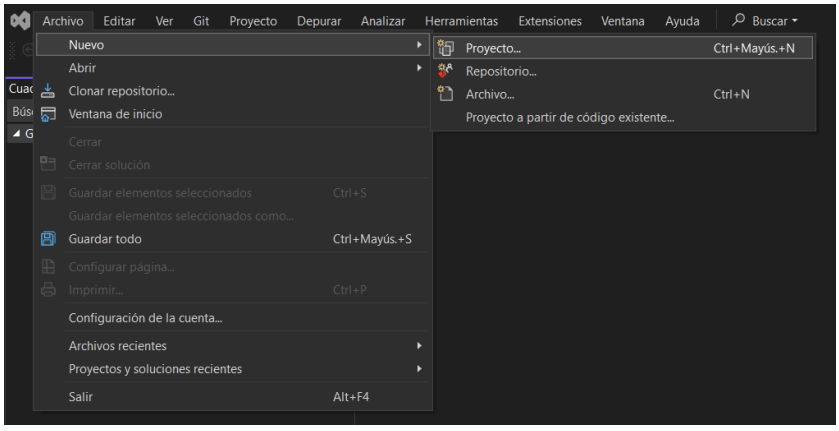
$P=V\times I$

P=V×I. La app valida la entrada, realiza el cálculo y muestra el resultado con formato numérico. Es una práctica ideal para aprender diseño de formularios, manejo de controles Label, TextBox y Button, y conversión/validación de datos en C#.

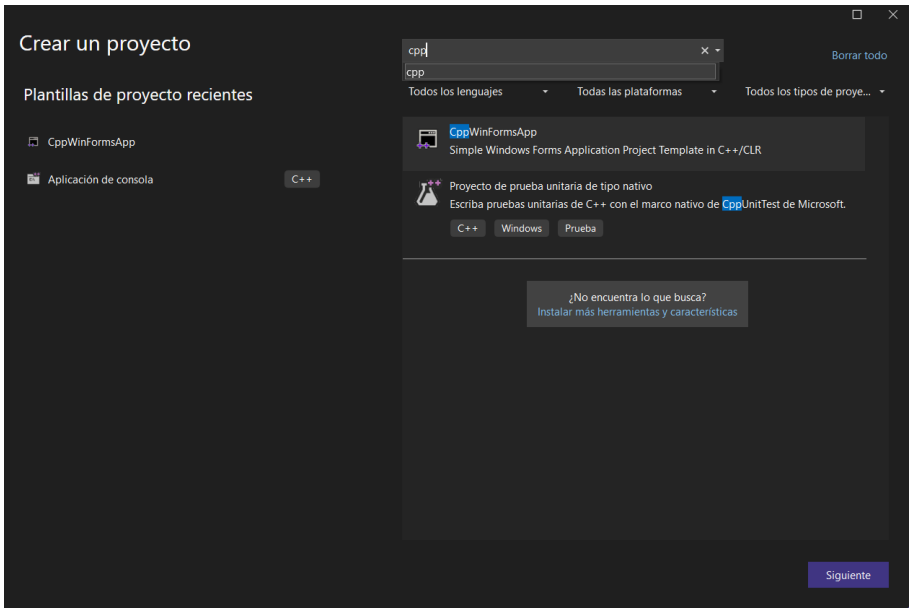
1) **Abrir visual studio 2022 previamente instalado**



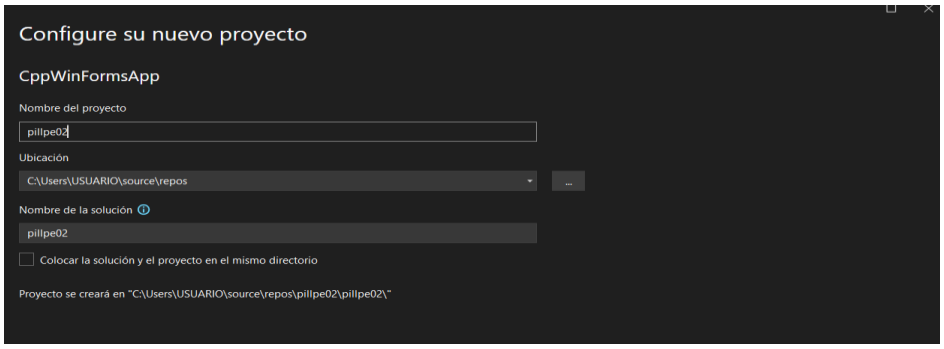
2) Selecciona un nuevo proyecto, entrando a archivo seleccionamos nuevo y al costado aparecerá proyecto nuevo presionamos esa opción y saldrá una ventana.



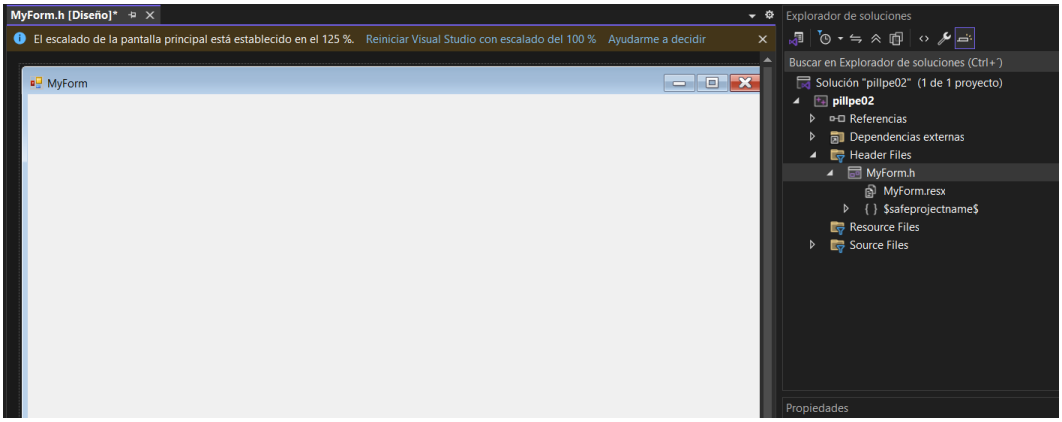
3) Crae un nuevo proyecto con cppWinFormsApp, buscado en el buscador cppWinFormsApp Previamnete instalada.



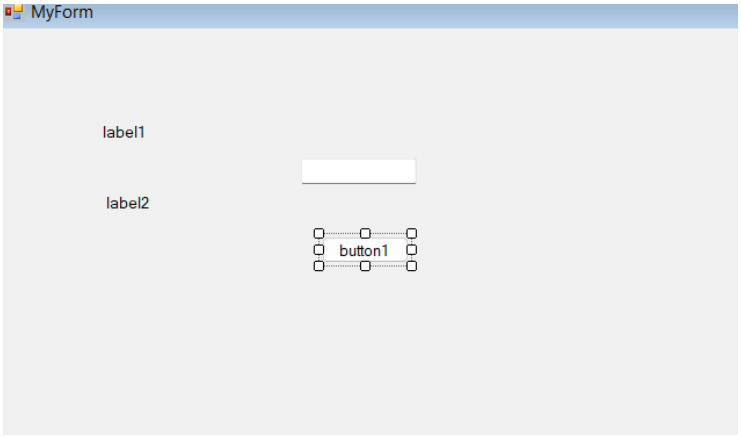
4) Configurar tu nuevo proyecto, donde escogeremos el nombre de nuestro proyecto



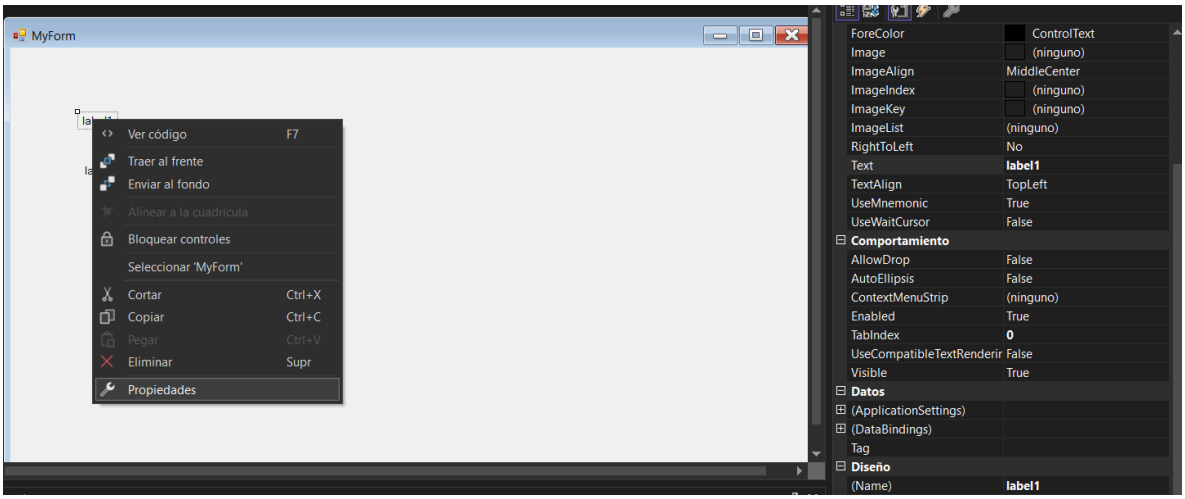
5) Abrir la pantalla blanca, en la mano derecha aparece headers files con dos clips presionamos Myform.h y en ese momento cargará y aparecerá la pantalla blanca



6) Ingresar los controles Label, TextBox y Button, a la pantalla blanca en la parte de arriba a la mano izquierda encontramos cuadro de herramientas en las opciones de ver donde selecionaremos las obciones label, texBox y Button



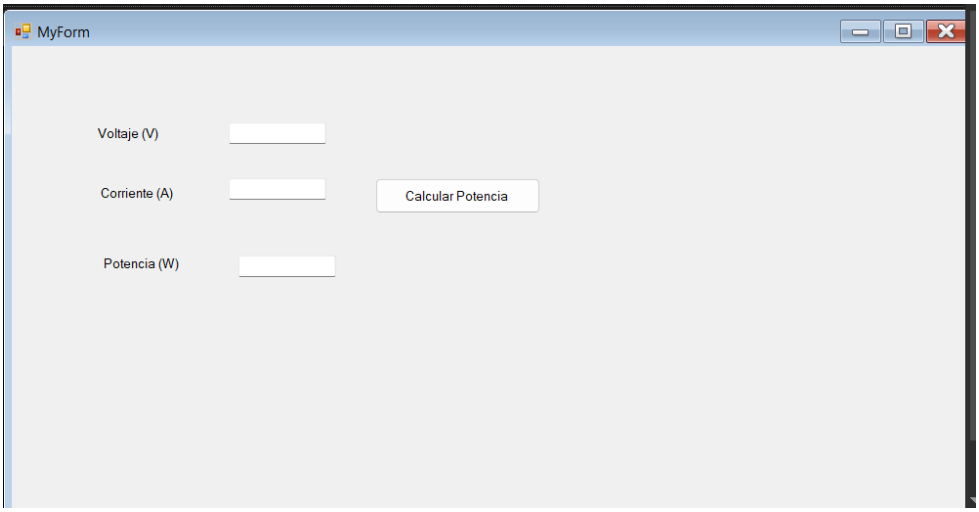
7) Editamos los controles label, TextBox y Button usando el mause dándole clip derecho donde se deplegara la función de configuración



8) Editamos los controles de label, TextBox y Button con los siguientes formula:

Tipo de Control	Nombre (Name)	Texto (Text)	Descripción
Label	lblVoltaje	Voltaje (V):	Indica dónde el usuario debe ingresar el voltaje.
TextBox	txtVoltaje	*(vacío)*	Campo donde se escribe el valor del voltaje.
Label	lblCorriente	Corriente (A):	Señala dónde escribir el valor de la corriente.
TextBox	txtCorriente	*(vacío)*	Campo donde se escribe la corriente eléctrica.
Button	btnCalcular	Calcular Potencia	Botón que ejecuta el cálculo al hacer clic.
Label	lblResultado	Potencia (W):	Indica el área donde se mostrará el resultado.
TextBox	txtResultado	*(vacío)* (ReadOnly)	Muestra el valor calculado. Está en solo lectura.

9)Una vez editado los controles se vera de la siguiente manera



10) Código para que funcione lo descrito en la pantalla:

```
#pragma once

namespace $safeprojectname$ {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
```

```
        {
                InitializeComponent();
                //
                //TODO: Add the constructor code here
                //
        }

protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~MyForm()
        {
                if (components)
                {
                        delete components;
                }
        }
private: System::Windows::Forms::Label^ lblVoltaje;
private: System::Windows::Forms::Label^ lblCorriente;
protected:

protected:

private: System::Windows::Forms::TextBox^ txtVoltaje;
private: System::Windows::Forms::Button^ btnCalcular;


private: System::Windows::Forms::TextBox^ txtCorriente;
```

```
        private: System::Windows::Forms::Label^ lblResultado;
        private: System::Windows::Forms::TextBox^ txtResultado;

        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                this->lblVoltaje = (gcnew System::Windows::Forms::Label());
                this->lblCorriente = (gcnew System::Windows::Forms::Label());
                this->txtVoltaje = (gcnew System::Windows::Forms::TextBox());
                this->btnCalcular = (gcnew System::Windows::Forms::Button());
                this->txtCorriente = (gcnew System::Windows::Forms::TextBox());
                this->lblResultado = (gcnew System::Windows::Forms::Label());
                this->txtResultado = (gcnew System::Windows::Forms::TextBox());
                this->SuspendLayout();
                //
                // lblVoltaje
                //
                this->lblVoltaje->AutoSize = true;
                this->lblVoltaje->Location = System::Drawing::Point(85, 81);
```

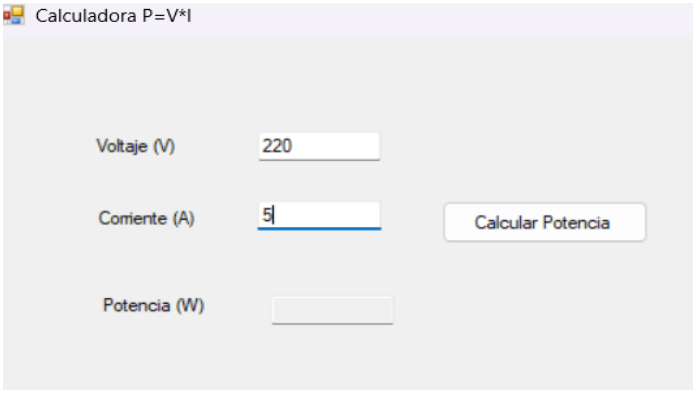
```
this->lblVoltaje->Name = L"lblVoltaje";
this->lblVoltaje->Size = System::Drawing::Size(69, 16);
this->lblVoltaje->TabIndex = 0;
this->lblVoltaje->Text = L"Voltaje (V)";
//
// lblCorriente
//
this->lblCorriente->AutoSize = true;
this->lblCorriente->Location = System::Drawing::Point(88, 142);
this->lblCorriente->Name = L"lblCorriente";
this->lblCorriente->Size = System::Drawing::Size(81, 16);
this->lblCorriente->TabIndex = 1;
this->lblCorriente->Text = L"Corriente (A)";
//
// txtVoltaje
//
this->txtVoltaje->Location = System::Drawing::Point(222, 78);
this->txtVoltaje->Name = L"txtVoltaje";
this->txtVoltaje->Size = System::Drawing::Size(100, 22);
this->txtVoltaje->TabIndex = 2;
//
// btnCalcular
//
this->btnCalcular->Location = System::Drawing::Point(372, 135);
this->btnCalcular->Name = L"btnCalcular";
this->btnCalcular->Size = System::Drawing::Size(169, 36);
this->btnCalcular->TabIndex = 3;
this->btnCalcular->Text = L"Calcular Potencia";
this->btnCalcular->UseVisualStyleBackColor = true;
```

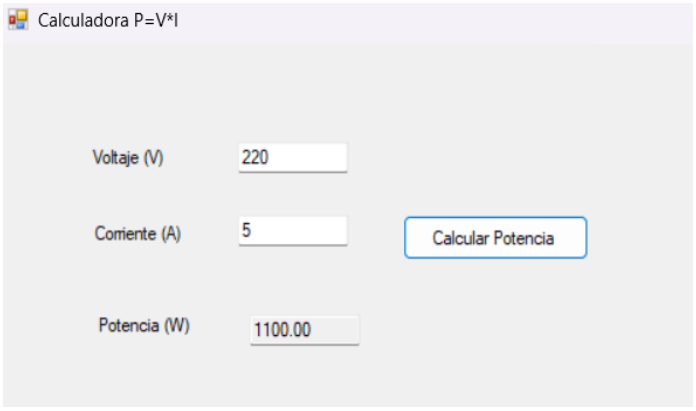
```
            this->btnCalcular->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
            //
            // txtCorriente
            //
            this->txtCorriente->Location = System::Drawing::Point(222, 135);
            this->txtCorriente->Name = L"txtCorriente";
            this->txtCorriente->Size = System::Drawing::Size(100, 22);
            this->txtCorriente->TabIndex = 4;
            //
            // lblResultado
            //
            this->lblResultado->AutoSize = true;
            this->lblResultado->Location = System::Drawing::Point(91, 214);
            this->lblResultado->Name = L"lblResultado";
            this->lblResultado->Size = System::Drawing::Size(84, 16);
            this->lblResultado->TabIndex = 5;
            this->lblResultado->Text = L"Potencia (W)";
            //
            // txtResultado
            //
            this->txtResultado->Location = System::Drawing::Point(232, 214);
            this->txtResultado->Name = L"txtResultado";
            this->txtResultado->ReadOnly = true; // El resultado no debe ser editable
            this->txtResultado->Size = System::Drawing::Size(100, 22);
            this->txtResultado->TabIndex = 6;
            //
            // MyForm
            //
```

```
            this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(980, 506);
            this->Controls->Add(this->txtResultado);
            this->Controls->Add(this->lblResultado);
            this->Controls->Add(this->txtCorriente);
            this->Controls->Add(this->btnCalcular);
            this->Controls->Add(this->txtVoltaje);
            this->Controls->Add(this->lblCorriente);
            this->Controls->Add(this->lblVoltaje);
            this->Name = L"MyForm";
            this->Text = L"Calculadora P=V*I";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
        // INICIO DE LA LÓGICA DE CÁLCULO
        try
        {
            // 1. Convertir los valores de texto (string) a números de punto flotante
(double)
            double voltaje = System::Convert::ToDouble(txtVoltaje->Text);
            double corriente = System::Convert::ToDouble(txtCorriente->Text);

            // 2. Aplicar la fórmula: Pin = V * I
            double potencia = voltaje * corriente;
```

```
                // 3. Mostrar el resultado en el TextBox de salida
                // Usamos "F2" para formatear el número con dos decimales para mayor claridad.
                this->txtResultado->Text = potencia.ToString("F2");
            }
            catch (System::FormatException^)
            {
                // Muestra una advertencia si el usuario no ingresa números válidos
                System::Windows::Forms::MessageBox::Show("Error: Ingrese valores numéricos válidos para Voltaje y Corriente.",
                    "Error de Datos",
System::Windows::Forms::MessageBoxButtons::OK,
System::Windows::Forms::MessageBoxIcon::Error);

                // Opcional: limpiar el campo de resultado y las entradas
                this->txtResultado->Text = "";
                this->txtVoltaje->Text = "";
                this->txtCorriente->Text = "";
            }
            // FIN DE LA LÓGICA DE CÁLCULO
        }
        };
}
```

11) Pondremos a prueba nuestro CALCULA LA POTENCIA ELÉCTRICA DE UN MOTOR, USANDO LA FÓRMULA BÁSICA





Conclusiones

Esta práctica permite al estudiante comprender la relación directa entre voltaje, corriente y potencia eléctrica, y al mismo tiempo aprender el diseño de interfaces gráficas en C#.

El ejercicio integra conceptos de programación estructurada, validación de datos y manejo de errores, aplicados a una situación real en el ámbito eléctrico.

El desarrollo de esta aplicación no solo refuerza los conocimientos de programación, sino también la capacidad de aplicar principios de la termodinámica y la electricidad en herramientas computacionales de cálculo.

Análisis de Corriente Continua y Corriente Alterna, y Desarrollo de Aplicación para Cálculo de Parámetros Eléctricos

PILLPE ROMANI FRANK

***Resumen*—Este informe presenta un análisis detallado de la Corriente Continua (CC) y Corriente Alterna (CA), explicando sus características fundamentales, aplicaciones prácticas en sistemas eléctricos y principios teóricos que las gobiernan. Además, se documenta el desarrollo de ejercicios prácticos realizados en entorno Windows Forms (C++/CLI), donde se implementaron cálculos eléctricos aplicando leyes fundamentales como Ley de Ohm, impedancia en circuitos RLC y validación de datos de entrada para evitar errores de formato. El objetivo del trabajo fue consolidar conocimientos teóricos mediante la implementación computacional precisa y confiable.**

***Abstract*-- This report presents a detailed analysis of Direct Current (DC) and Alternating Current (AC), explaining their fundamental characteristics, practical applications in electrical systems, and the theoretical principles that govern them. Furthermore, it documents the development of practical exercises performed in a Windows Forms environment (C++/CLI), where electrical calculations were implemented applying fundamental laws such as Ohm's Law, impedance in RLC circuits, and input data validation to prevent formatting errors. The objective of this work was to consolidate theoretical knowledge through precise and reliable computational implementation.**

I. Introducción

La electricidad se ha convertido en un recurso indispensable para la operación de sistemas de comunicación, transporte, automatización industrial, informática y equipamiento doméstico. Comprender su comportamiento no solo es esencial desde el punto de vista académico, sino también desde la perspectiva de ingeniería aplicada [1].

Entre las principales formas de suministro eléctrico se encuentran la Corriente Continua y la Corriente Alterna. Cada una presenta ventajas específicas dependiendo del uso, condiciones de operación y requerimientos técnicos de los dispositivos que las utilizan [2].

La Corriente Continua se emplea ampliamente en electrónica, almacenamiento energético y sistemas alimentados por baterías, mientras que la Corriente Alterna es predominante en sistemas eléctricos de potencia debido a su facilidad de generación, transformación y transmisión a grandes distancias [1][3].

En este trabajo, además del análisis teórico, se integró una herramienta computacional que permite automatizar cálculos eléctricos, reforzando el aprendizaje y asegurando resultados precisos mediante validación de datos y correcta implementación matemática [4].

II. FUNDAMENTOS DE CORRIENTE CONTINUA (CC)

A. *Definición*

La Corriente Continua se caracteriza por mantener una dirección de circulación constante y un valor que puede ser fijo o variar lentamente sin invertir su polaridad. Este tipo de corriente es típico de fuentes químicas y electrónicas como baterías, paneles solares y fuentes reguladas [1][2].

B. *Fuentes de Corriente Continua*

Entre sus principales fuentes destacan las baterías recargables, pilas, celdas solares y fuentes electrónicas rectificadas. Su estabilidad hace que sea ampliamente utilizada en dispositivos electrónicos, telecomunicaciones, control industrial y sistemas digitales [1][4].

C. *Ley de Ohm en CC*

El análisis de circuitos de CC se fundamenta principalmente en la Ley de Ohm:

$$V=I\cdot R$$

Esta relación permite establecer dependencias directas entre voltaje, corriente y resistencia, convirtiéndose en herramienta base para diseño eléctrico y diagnóstico de fallas [1][2].

D. *Circuitos Resistivos*

En los circuitos analizados dentro del proyecto, se abordaron configuraciones en serie y paralelo. En serie la resistencia equivalente se obtiene sumando las resistencias individuales, mientras que en paralelo se emplea la relación:

$$\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

Estos cálculos fueron implementados correctamente en la aplicación desarrollada, demostrando la coherencia entre teoría y práctica [2][4].

III. FUNDAMENTOS DE CORRIENTE ALTERNA (CA)

A. Naturaleza de la CA

La Corriente Alterna se caracteriza por variar su magnitud y sentido de forma periódica, normalmente representada mediante una onda sinusoidal. Este tipo de suministro es utilizado en la distribución eléctrica global debido a su eficiencia para transmisión a larga distancia y su capacidad para modificar niveles de voltaje mediante transformadores [1][3].

B. Parámetros Característicos

Entre los parámetros fundamentales destacan la frecuencia, el periodo y la velocidad angular:

$\omega=2\pi f$

Estos parámetros determinan el comportamiento dinámico de los circuitos eléctricos y su respuesta frente a diferentes cargas [2][3].

C. Circuitos RLC

En la aplicación se trabajó con circuitos compuestos por resistencia (R), inductancia (L) y capacitancia (C), elementos que introducen oposición al cambio de corriente llamada reactancia [1][4].

Las reactancias se definen como:

$$X_L = 2\pi f L$$

$$X_C = \frac{1}{2\pi f C}$$

La impedancia total se calcula mediante:

$$Z = \sqrt{R^2 + (X_L - X_C)^2}$$

y la corriente resultante:

$$I = \frac{V}{Z}$$

Estos cálculos permitieron analizar correctamente el comportamiento del sistema en régimen sinusoidal [1][2][4].

IV. DESARROLLO DE LA APLICACIÓN

La aplicación desarrollada en Windows Forms con C++/CLI tuvo como finalidad facilitar cálculos eléctricos, mejorar la interpretación de resultados y servir como apoyo académico. Se priorizó la robustez, validación de datos y facilidad de uso [4].

A. Objetivos Funcionales

- Calcular parámetros en circuitos de CC y CA.
- Automatizar cálculos evitando errores manuales.
- Aceptar formato decimal con punto o coma.
- Mostrar mensajes claros ante datos incorrectos.

B. Resolución de Problemas Técnicos

Durante el desarrollo se presentaron complicaciones como funciones duplicadas, variables no declaradas y errores de conversión numérica. Estos inconvenientes fueron solucionados mediante correcta organización del código y aplicación de buenas prácticas recomendadas en literatura especializada [1][3][4].

V. RESULTADOS

El software desarrollado permite realizar cálculos confiables, gestionar adecuadamente datos de entrada, evitar errores por formato y ofrecer información estructurada al usuario. Desde el punto de vista académico, el proyecto reforzó la comprensión de fenómenos eléctricos y su traducción a lógica computacional, logrando integración exitosa entre teoría y práctica [1][2][4].

VI.CONCLUSION

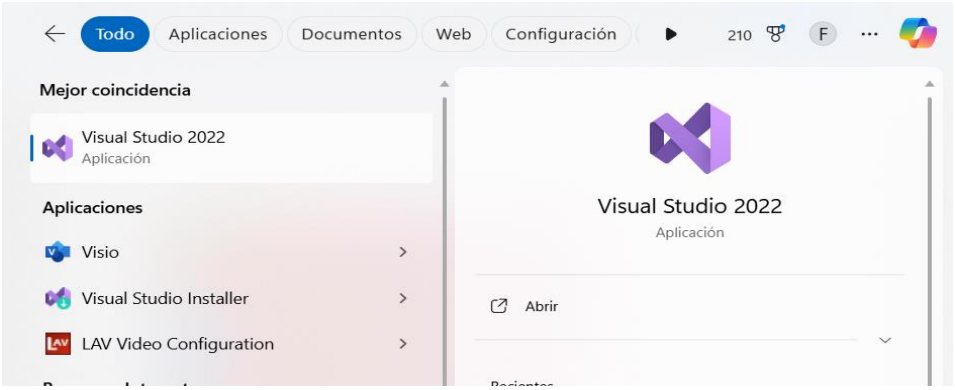
La comprensión de Corriente Continua y Alterna es esencial para cualquier área relacionada con ingeniería eléctrica. El proyecto permitió consolidar conceptos fundamentales, aplicar leyes eléctricas, analizar circuitos reales y, al mismo tiempo, desarrollar una herramienta informática útil y eficiente. Se evidencia que la implementación computacional fortalece el aprendizaje y contribuye a la precisión técnica [1][2][3].

REFERENCIAS

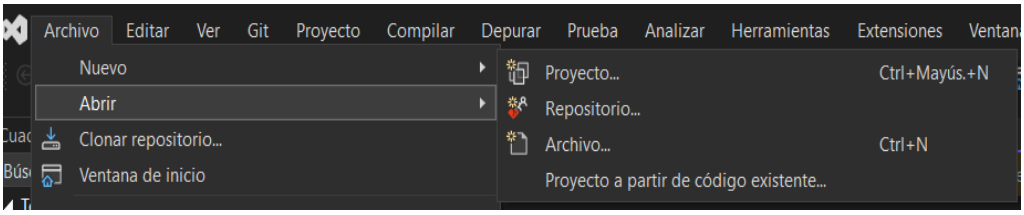
[1] [1] A. Sadiku, Fundamentals of Electric Circuits, McGraw-Hill, 2015.
[2] [2] R. Boylestad, Introductory Circuit Analysis, Pearson, 2013.
[3] [3] IEEE Standards Association, “Electrical Terminology Guidelines,” IEEE, 2020.
[4] [4] J. Nilsson and S. Riedel, Electric Circuits, Pearson, 2019.

MENÚ DINÁMICO

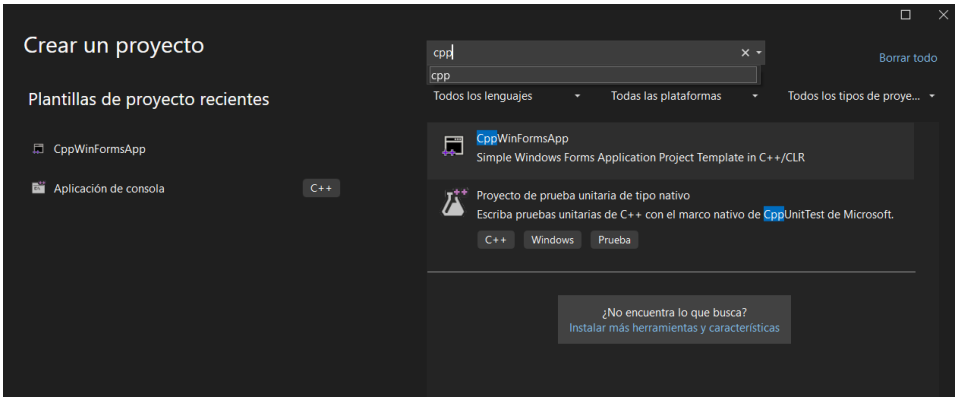
1. **Buscamos en el buscador visual studio2022**



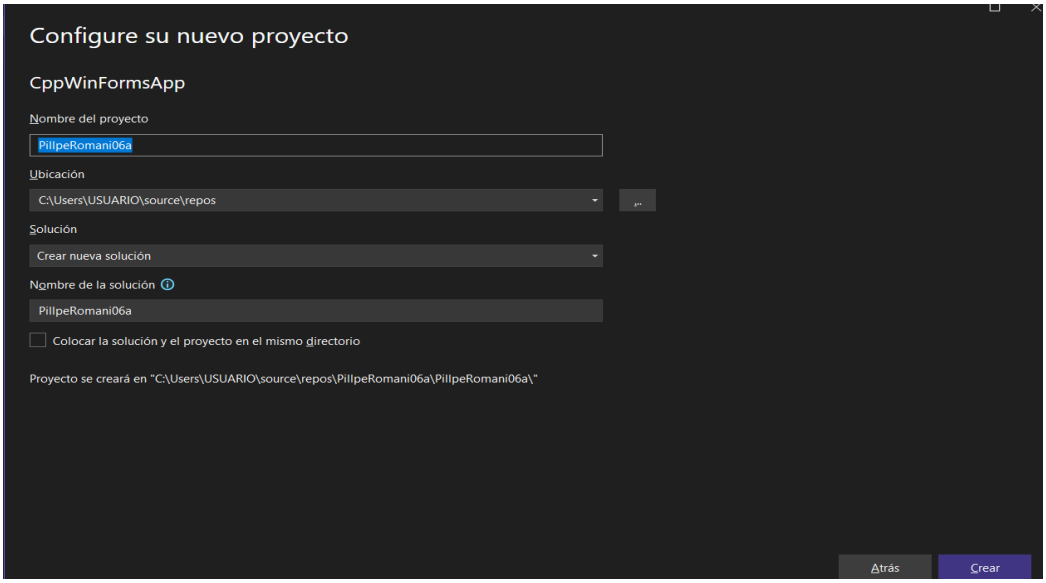
2. **Creamos un nuevo proyecto nuevo**



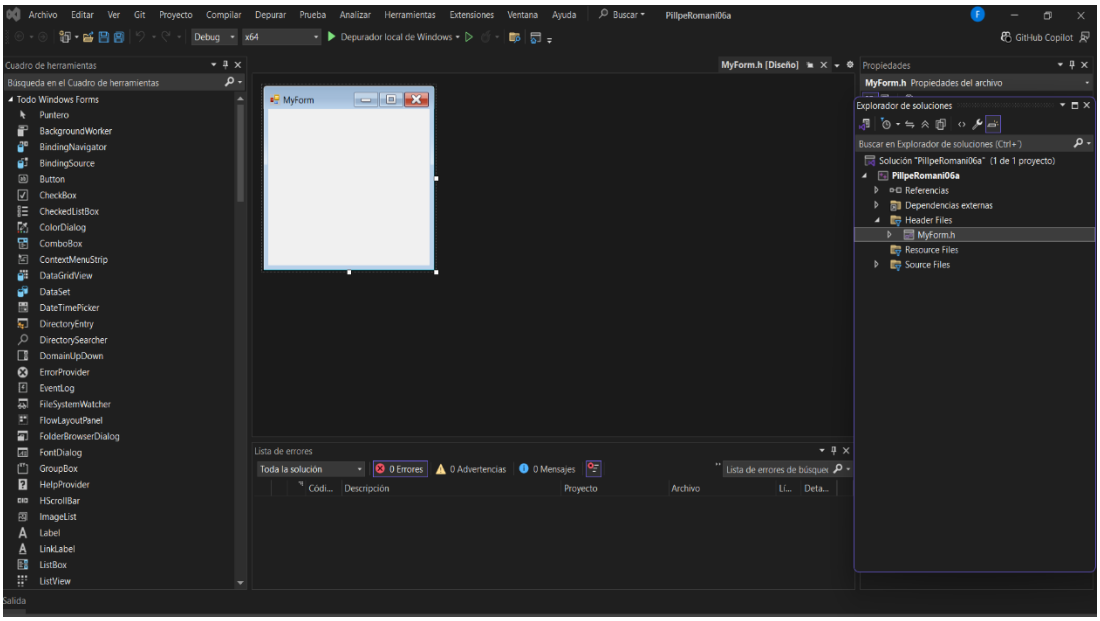
3. **Seleccionamos cpp**



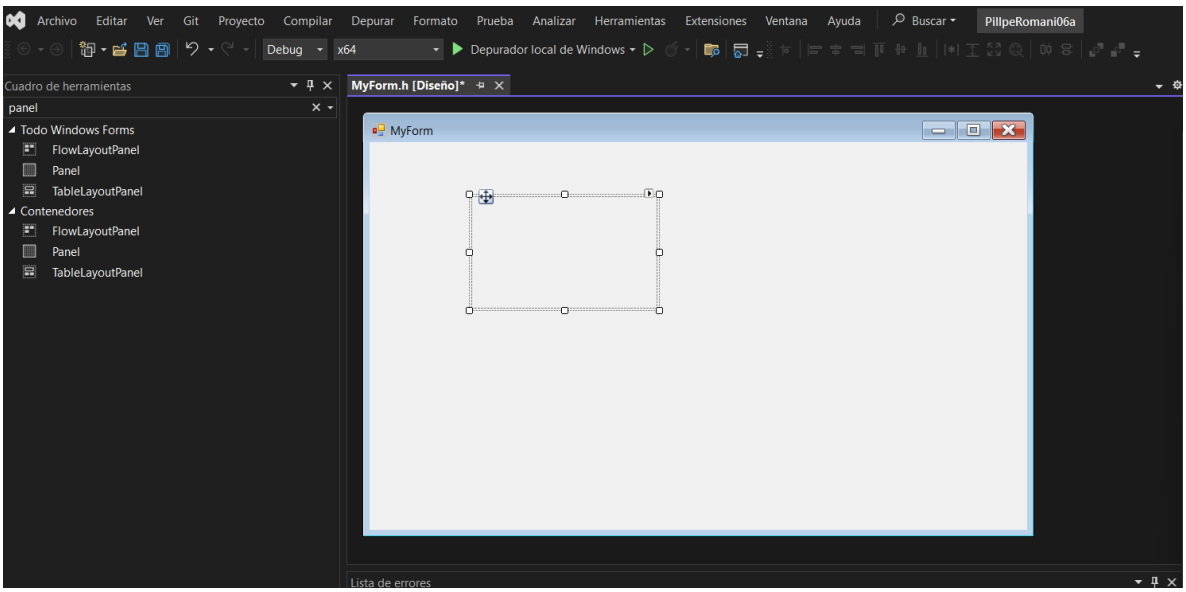
4. **Colocar nuestros apellidos seguido de la clase y el intento**



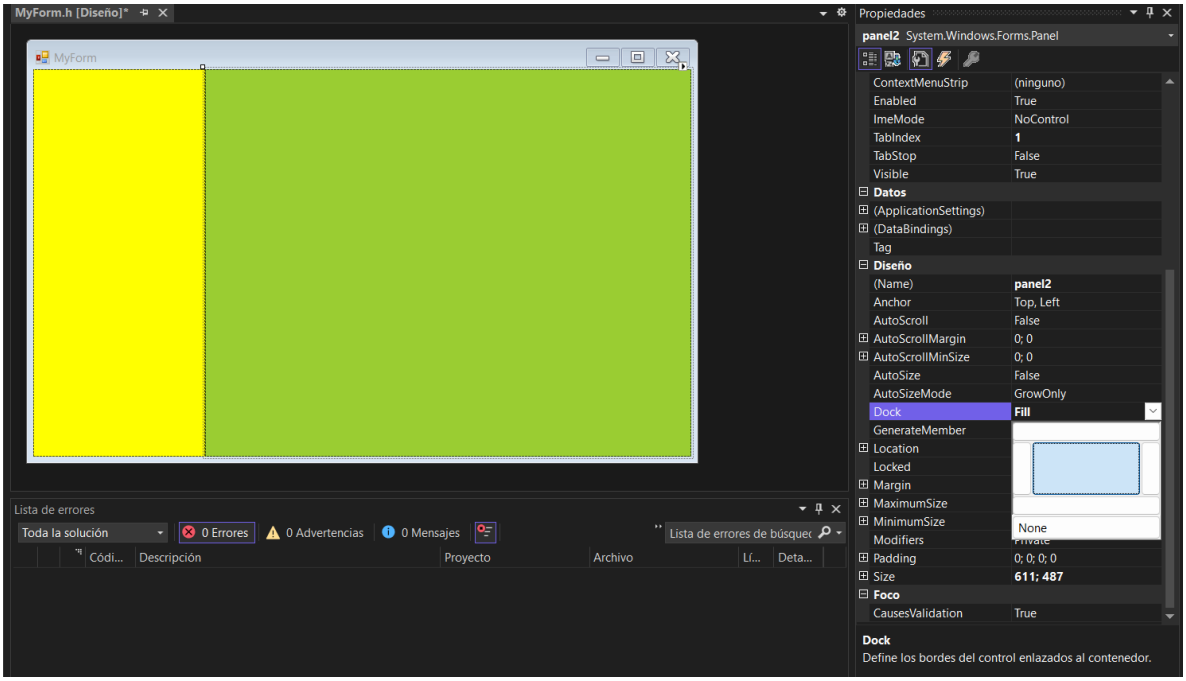
5. **Agregamos el interfaz principal, ubicando el explorador de soluciones buscando la opción de MyForm.h**



6. Agregamos las opciones de panel

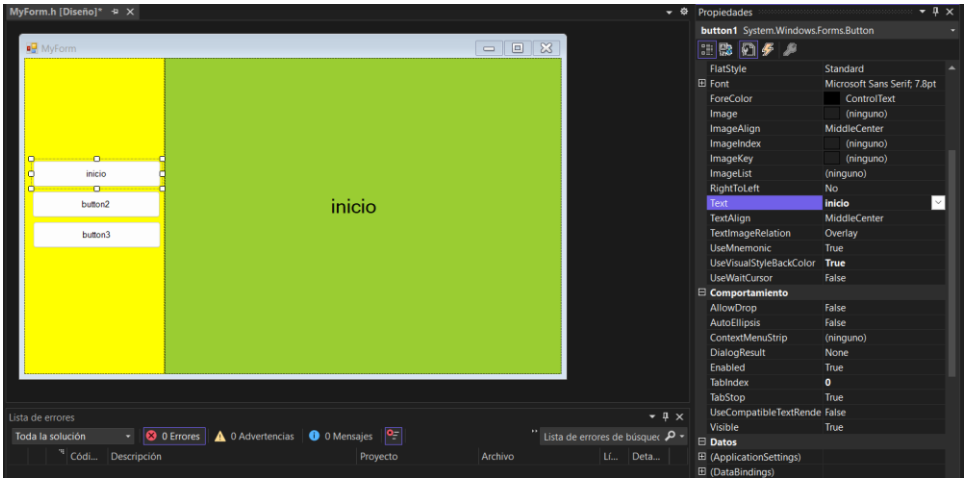


7. Cambiamos el color en backcolor y ubicamos la proporción del panel con Dock

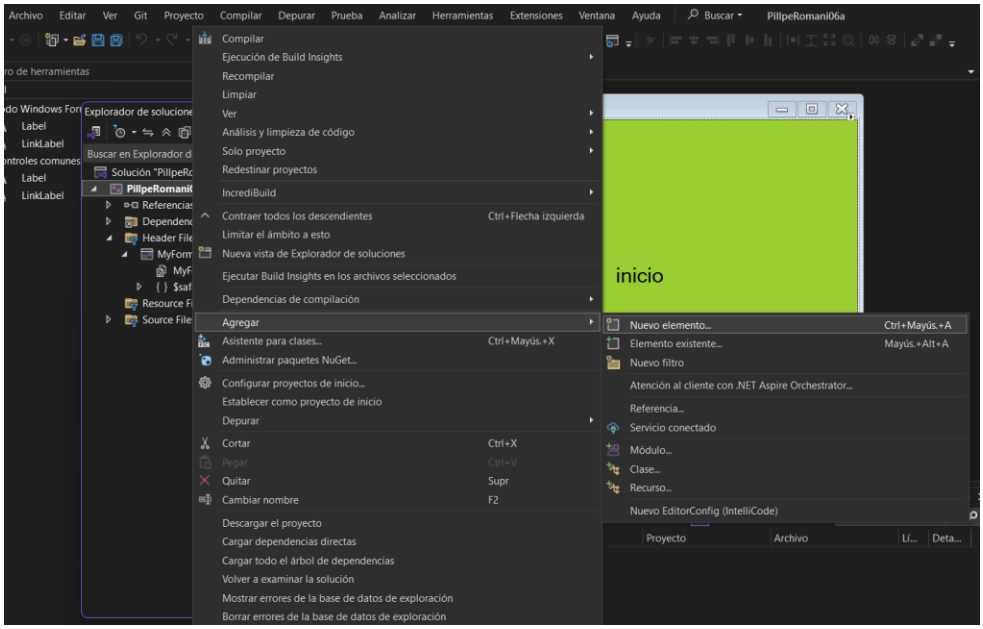


8. Agregamos el label y button

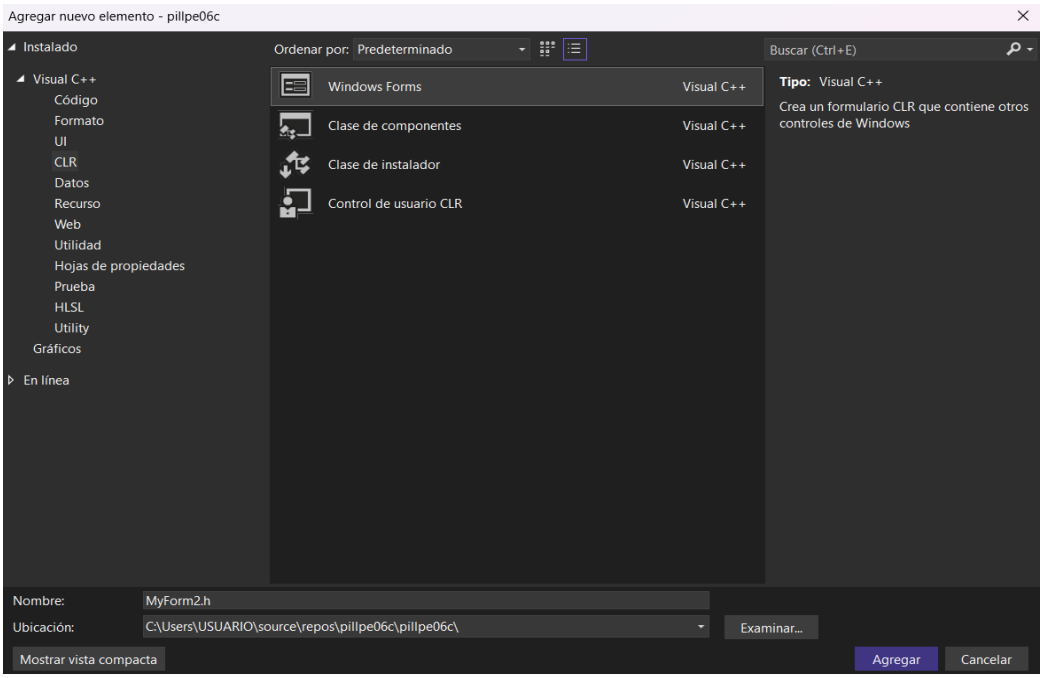
9. Modificamos los nombres en la opción de text



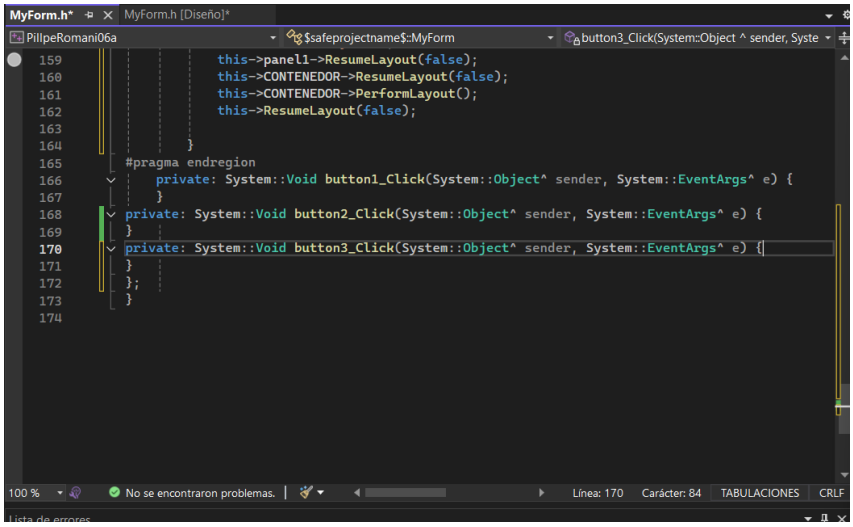
10. **Buscar en explorador de soluciones el nombre del archivo, con click derecho ubicamos la opción de agregar y nuevo elemento**



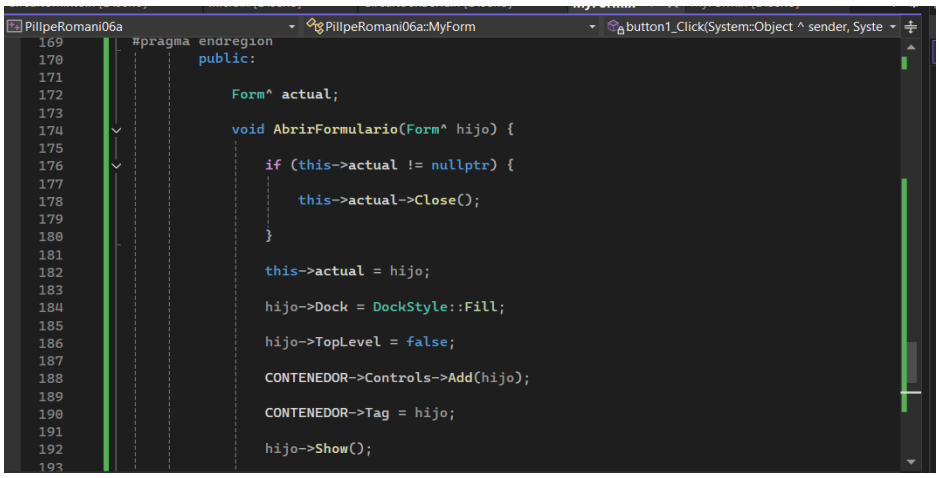
11. **Creamos nuestras ventanas**

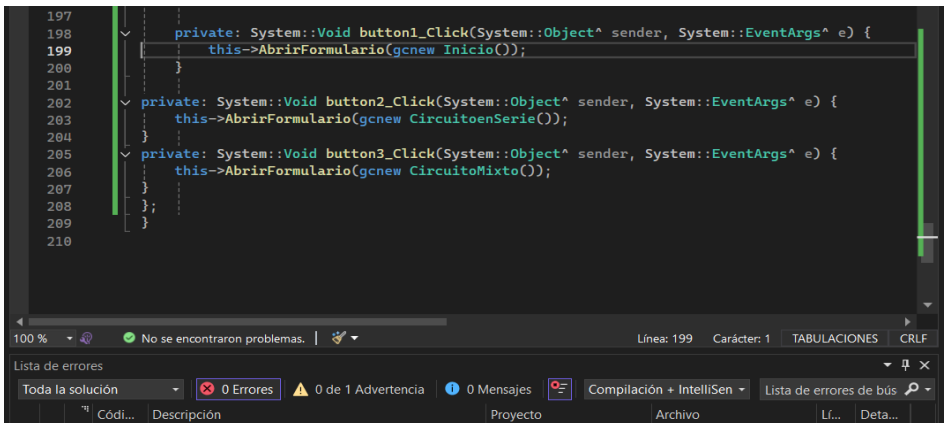


12. Entramos al código haciendo doble click a los tres button

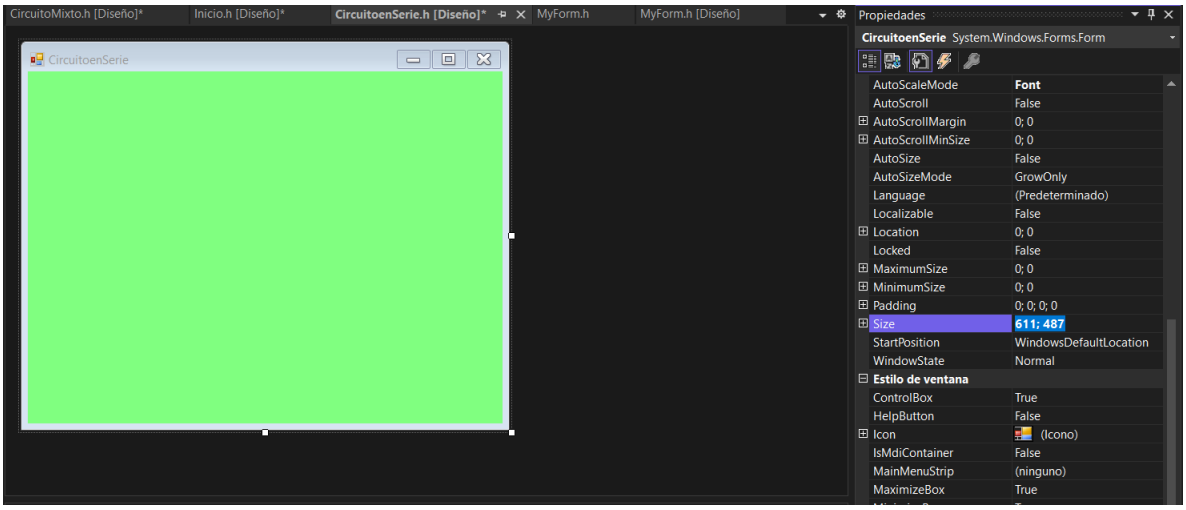


13. Agregar los códigos para poder usar las ventanas y estén definidos como va a funcionar nuestro menú dinámico

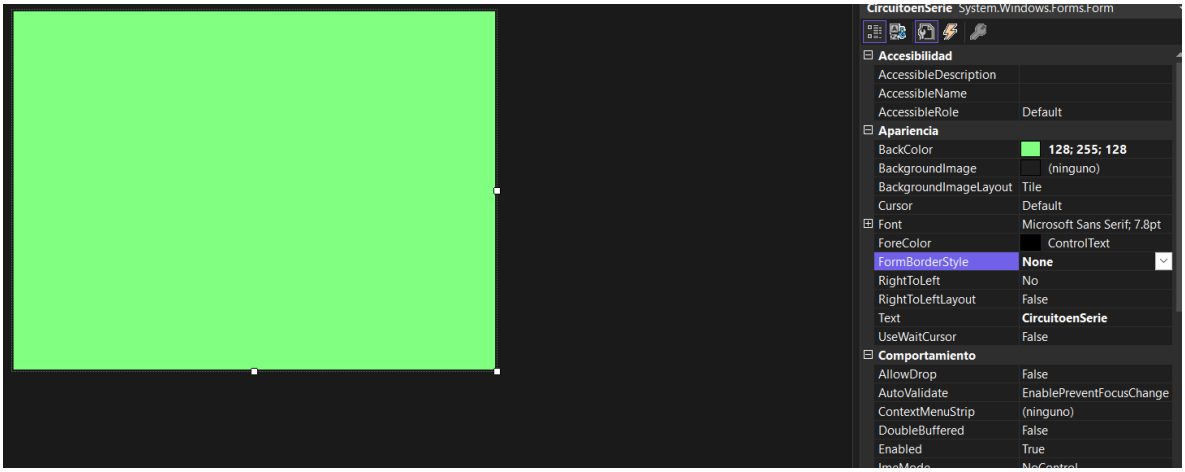




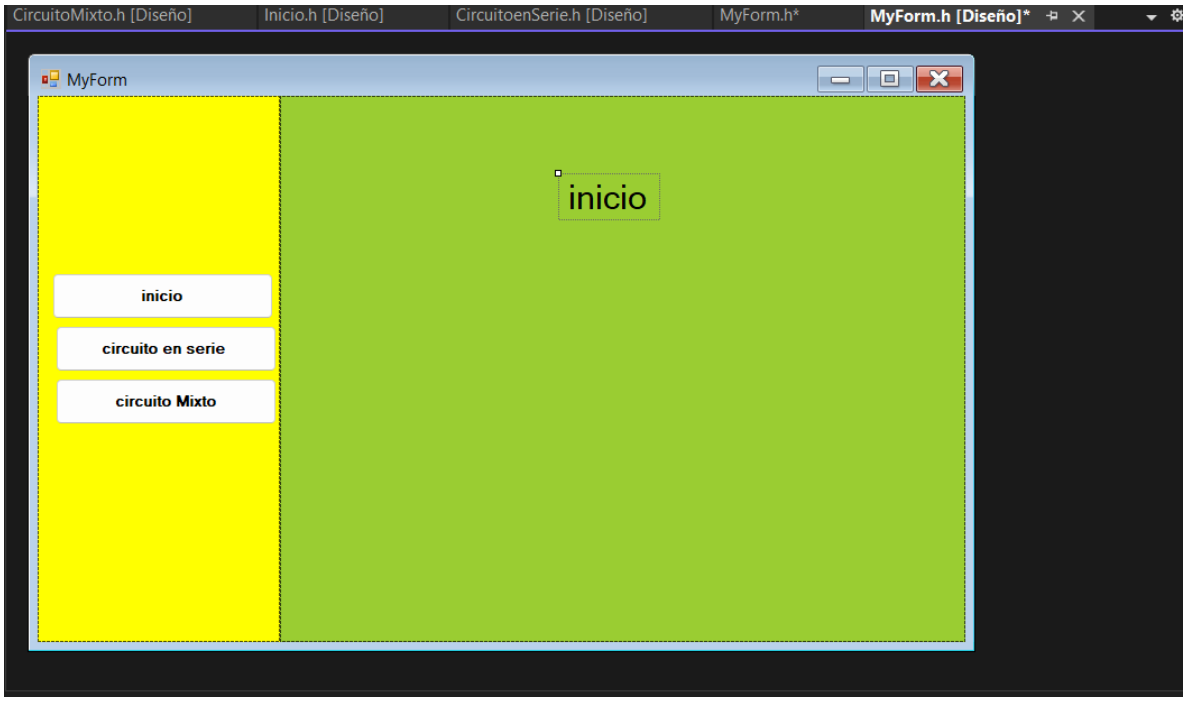
14. Ponemos todas las ventanas con el mismo tamaño copiando el size del panel principal



15. Cambiamos el interfaz cambiando el FormBorderSytle a none



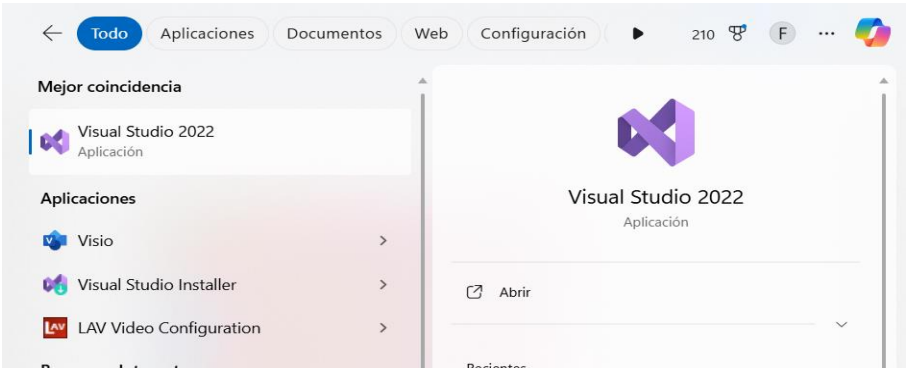
16. **Lo depuramos y quedara de la siguiente manera**



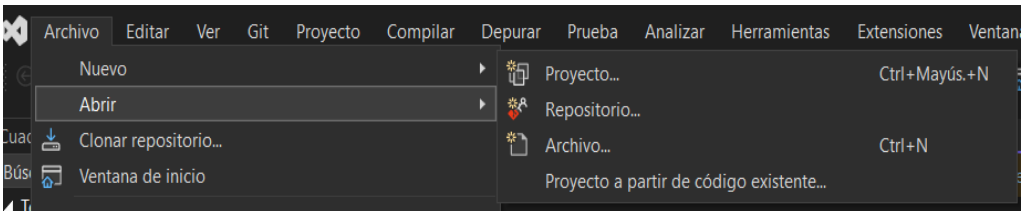
EVAP07

ABRIR UN SEGUNDO FORMULARIO

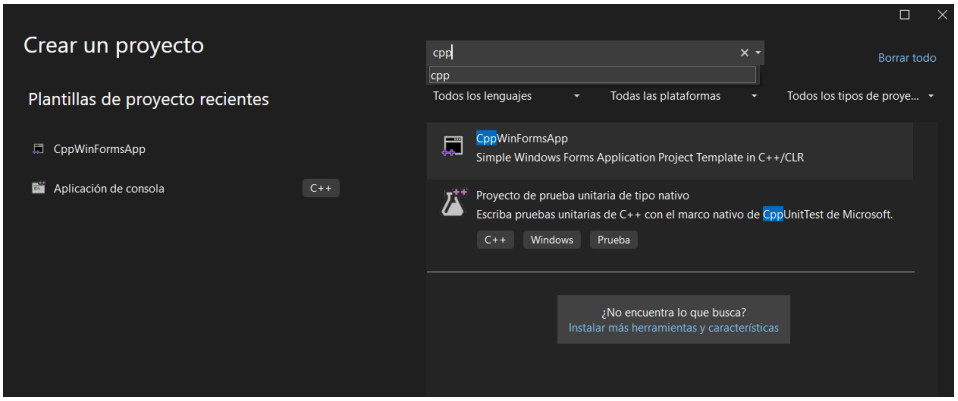
1. **Buscamos en el buscador visual studio2022.**



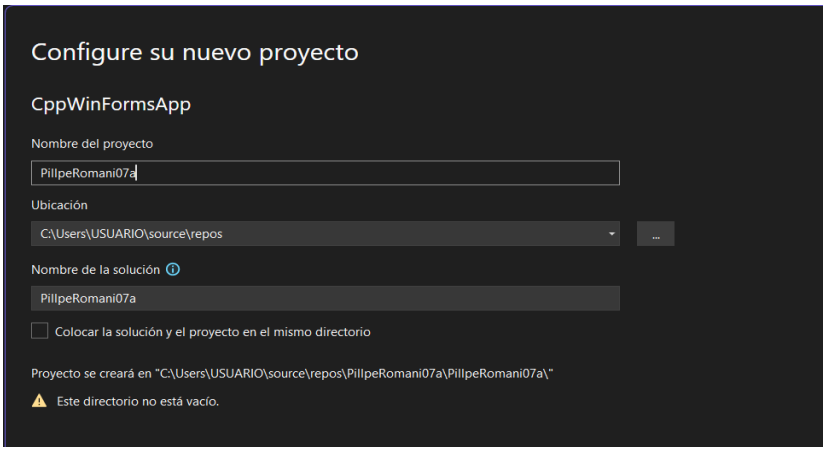
2. **Creamos un nuevo proyecto nuevo.**



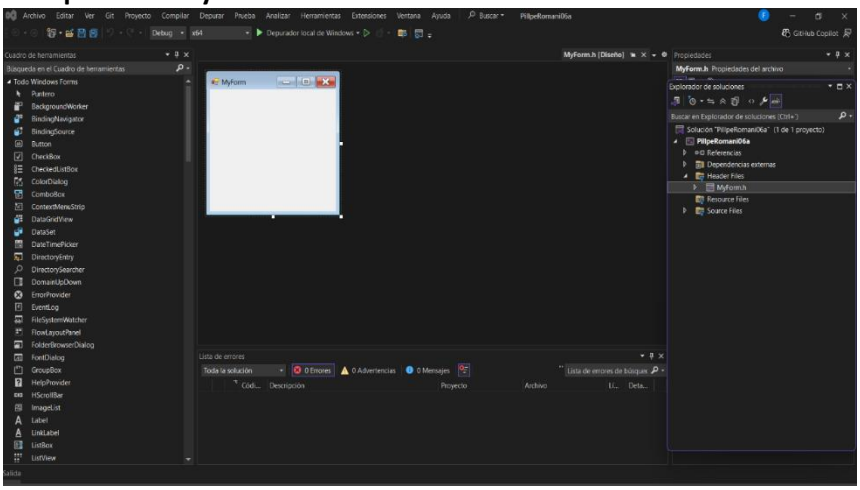
3. **Seleccionamos cpp.**



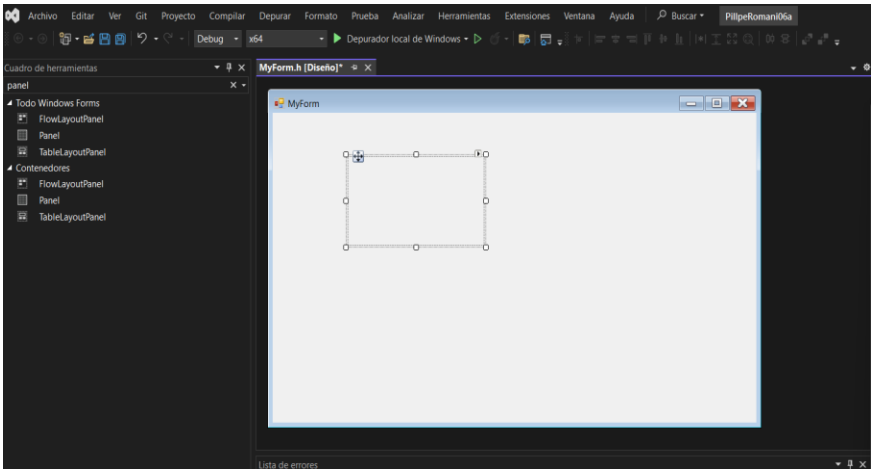
4. **Colocar nuestros apellidos seguido de la clase y el intento.**



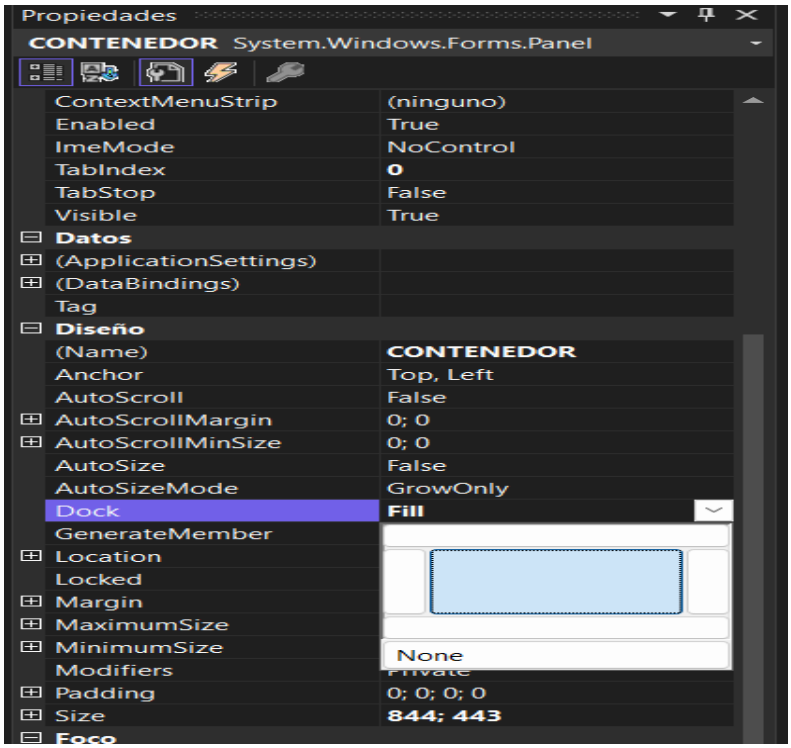
5. **Agregamos el interfaz principal, ubicando el explorador de soluciones buscando la opción de MyForm.h**



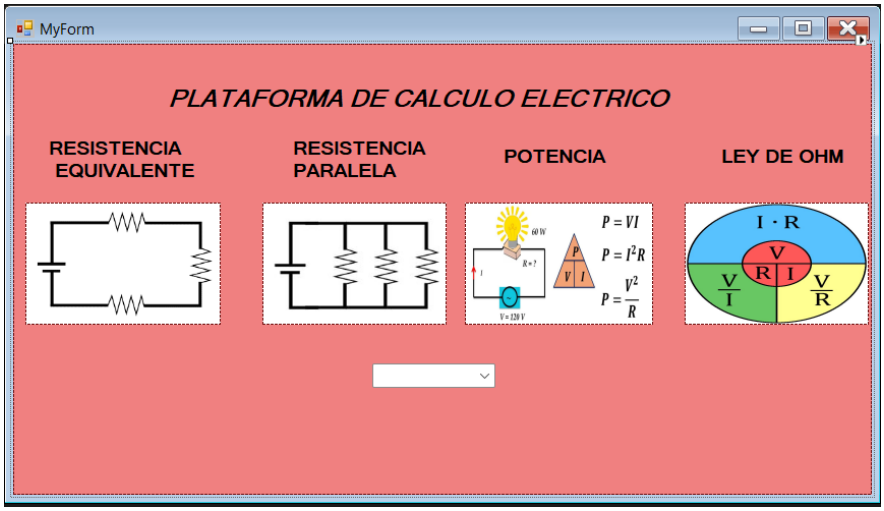
6. **Agregamos las opciones de panel.**



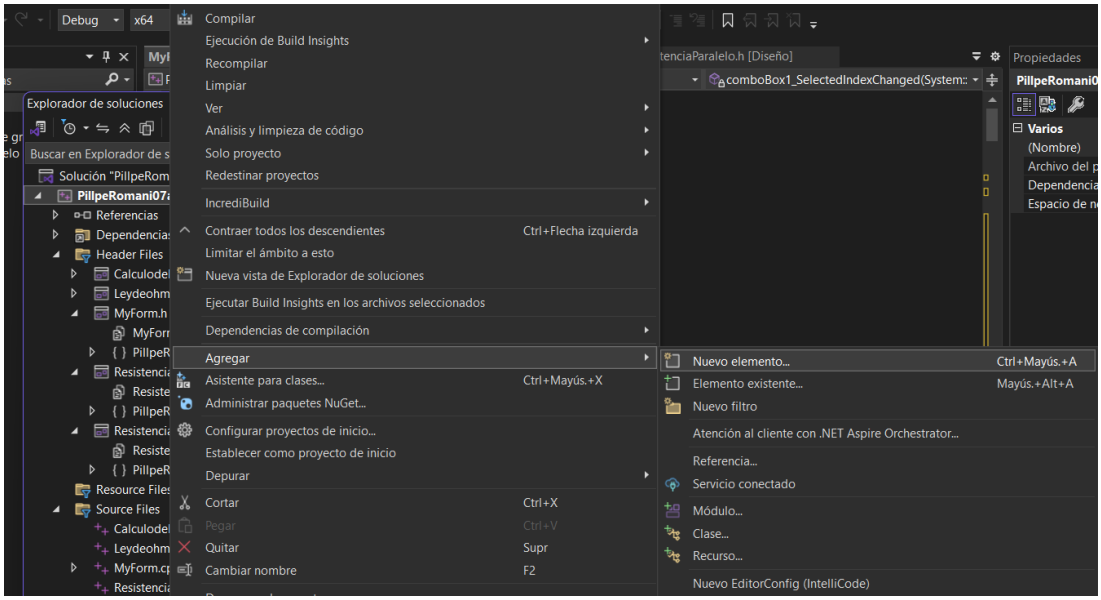
7. **Modificamos el panel cambiando el DOCK a fill y el NAME a CONTENEDOR.**



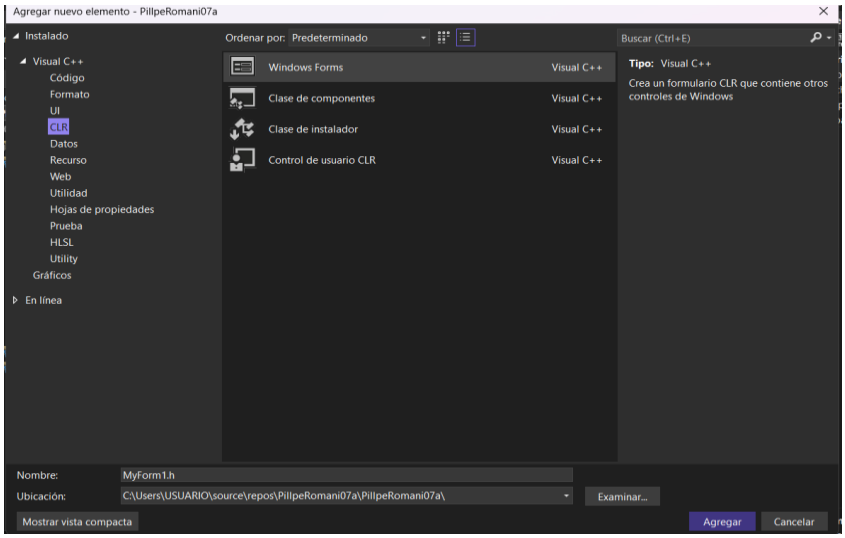
8. **Agregar los label, picturebox y combobox. Los label darán nombre, picterebux será para las imágenes y el combobox para hacer las ventanas secundarias quedaría de la siguiente manera.**



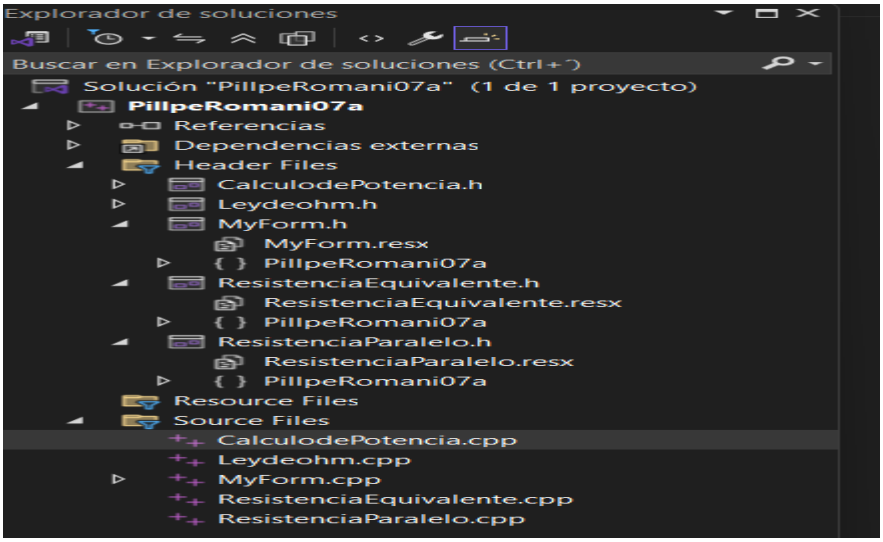
9. **Agregamos los formularios ubicando el nombre de nuestro archivo dándole clic derecho, en agregar y seleccionar nuevo elemento.**



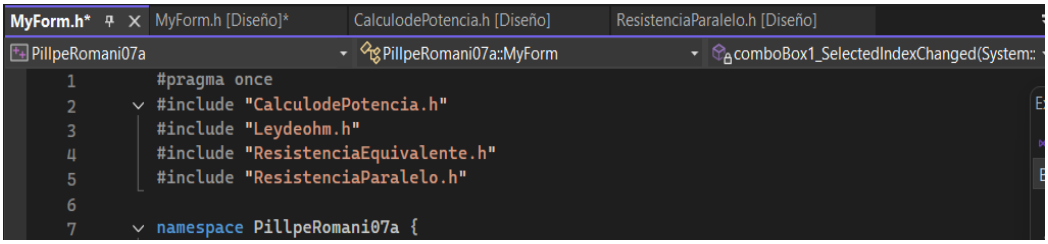
10. **Escogemos el CLR en windowsforms y en nombre colocamos el tema de nuestro formulario.**



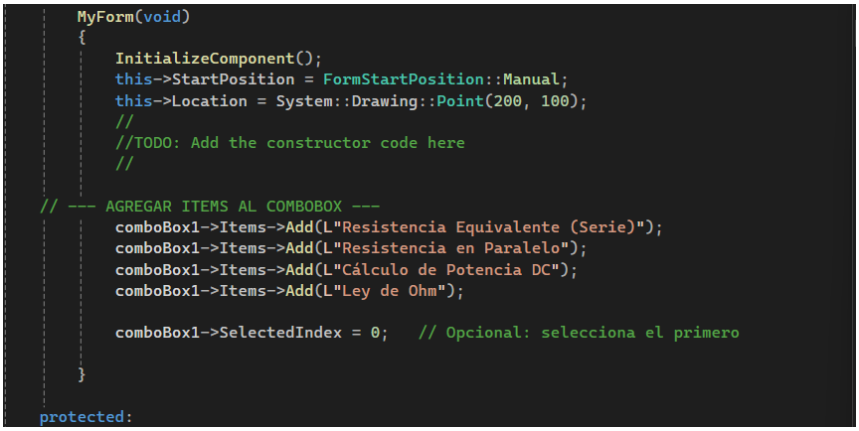
11. **Colocamos cada código de cpp de los formularios en el código de la interfaz principal.**



12. **Quedaría de la siguiente manera en el código de la interfaz principal.**



13. **Agregamos los siguientes códigos debajo de initializecomponent para mantener todos los formularios en una misma posición el otro código agregara los ítems en el COMBOBOX que también se puede poner de manera manual en las opciones de COMBOBOX.**



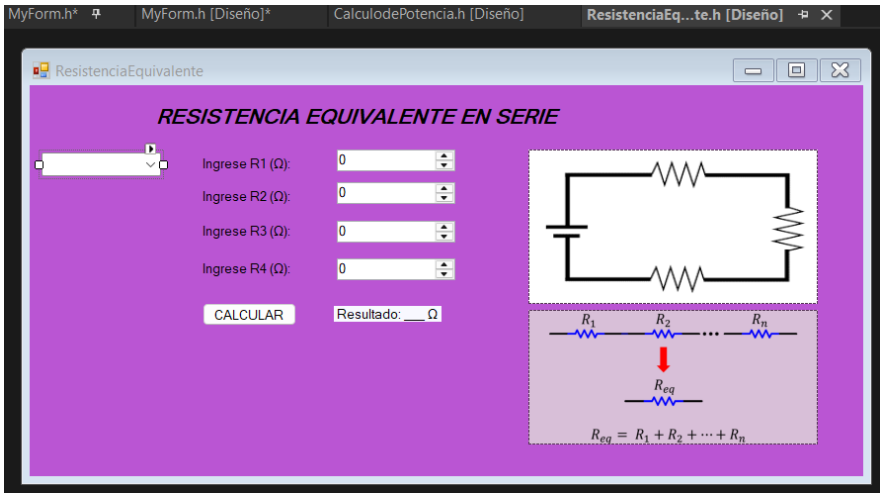
```
    MyForm(void)
    {
        InitializeComponent();
        this->StartPosition = FormStartPosition::Manual;
        this->Location = System::Drawing::Point(200, 100);
        //
        //TODO: Add the constructor code here
        //

// --- AGREGAR ITEMS AL COMBOBOX ---
        comboBox1->Items->Add(L"Resistencia Equivalente (Serie)");
        comboBox1->Items->Add(L"Resistencia en Paralelo");
        comboBox1->Items->Add(L"Cálculo de Potencia DC");
        comboBox1->Items->Add(L"Ley de Ohm");

        comboBox1->SelectedIndex = 0;   // Opcional: selecciona el primero

    }

protected:
```

14. Código para que funcione el COMBOBOX y pueda desplegar los formularios

```
247     private: System::Void comboBox1_SelectedIndexChanged(System::Object^ sender, System::Even
248
249         Form^ formulario;
250
251         if (comboBox1->SelectedIndex == 0)
252             formulario = gcnew ResistenciaEquivalente();
253         else if (comboBox1->SelectedIndex == 1)
254             formulario = gcnew ResistenciaParalelo();
255         else if (comboBox1->SelectedIndex == 2)
256             formulario = gcnew CalculodePotencia();
257         else
258             formulario = gcnew Leydeohm();
259
260         // OCULTA EL FORMULARIO PRINCIPAL
261         this->Hide();
262
263         // MUESTRA SOLO EL FORMULARIO SELECCIONADO
264         formulario->ShowDialog();
265
266         // CUANDO SE CIERRE EL FORMULARIO, VUELVE A MOSTRAR EL PRINCIPAL (opcional)
267
268         this->Show();
269
270     }
```

15. Modificamos los formularios iniciamos con el de resistencia equivalente agregando

a. **LABEL: nombre del tirulo y las resistencias**
b. **BUTTON: para poder calcular**
c. **NUMERICUPDOWN: para agregar los números**
d. **PICTUREBOX: imágenes referenciales**
e. **CONBOBOX: escoger cuantas resistencias usar**



16. Código para el combobox

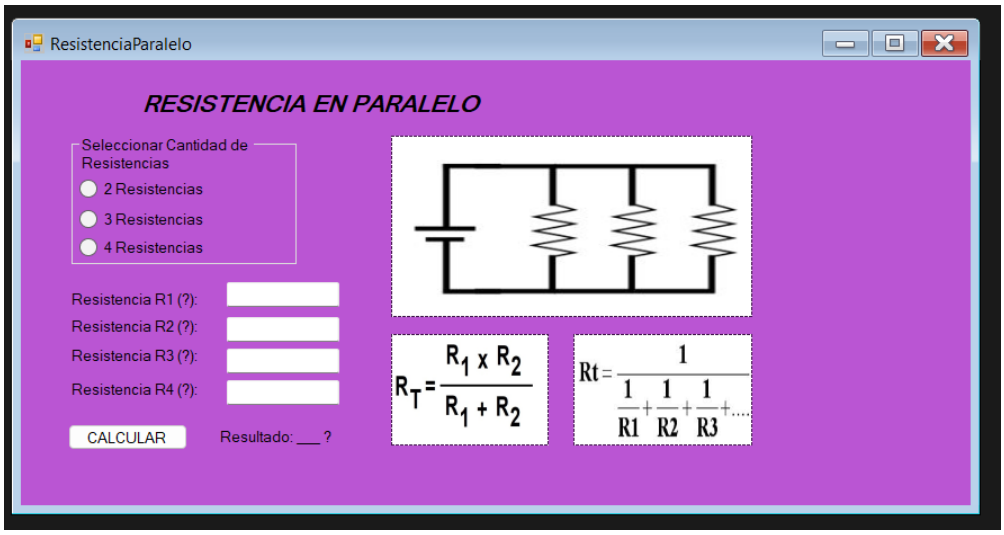
```
private: System::Void cbCantidadResistencias_SelectedIndexChanged(System::Object^ sender
    // Ocultamos todo primero
    numR1->Visible = true;
    numR2->Visible = true;
    numR3->Visible = false;
    numR4->Visible = false;

    labelR1->Visible = true;
    labelR2->Visible = true;
    labelR3->Visible = false;
    labelR4->Visible = false;

    // 2 resistencias
    if (cbCantidadResistencias->SelectedIndex == 0) {
        numR3->Visible = false;
        numR4->Visible = false;
        labelR3->Visible = false;
        labelR4->Visible = false;
    }

    // 3 resistencias
    else if (cbCantidadResistencias->SelectedIndex == 1) {
        numR3->Visible = true;
        labelR3->Visible = true;
        numR4->Visible = false;
```

```
    // 3 resistencias
    else if (cbCantidadResistencias->SelectedIndex == 1) {
        numR3->Visible = true;
        labelR3->Visible = true;
        numR4->Visible = false;
        labelR4->Visible = false;
    }

    // 4 resistencias
    else {
        numR3->Visible = true;
        numR4->Visible = true;
        labelR3->Visible = true;
        labelR4->Visible = true;
    }
}
```

17. Código para el buttun calcular.

```
private: System::Void btnCalcular_Click(System::Object^ sender, System::EventArgs^ e) {
    double R1 = (double)numR1->Value;
    double R2 = (double)numR2->Value;
    double R3 = (double)numR3->Value;
    double R4 = (double)numR4->Value;

    double Req = 0;

    // 2 resistencias
    if (cbCantidadResistencias->SelectedIndex == 0)
        Req = R1 + R2;

    // 3 resistencias
    else if (cbCantidadResistencias->SelectedIndex == 1)
        Req = R1 + R2 + R3;

    // 4 resistencias
    else
        Req = R1 + R2 + R3 + R4;

    labelResultado->Text = "Resultado: " + Req.ToString() + " Ω";

}
```

18. **Pasmos al formulario de resistencia paralelo**
    a. **LABEL: para los nombres del título, las resistencias y el resultado**
    b. **BUTTUN: para poder calcular**
    c. **PICTERUBOX: para agregar imágenes**
    d. **GROUPBOX: para agrupar a un cuadro para un grupo de controles**
    e. **RADIOBUTTUN: las opciones de cuantas resistencias usar**
    f. **TEXBOX: poner los números al calcular**

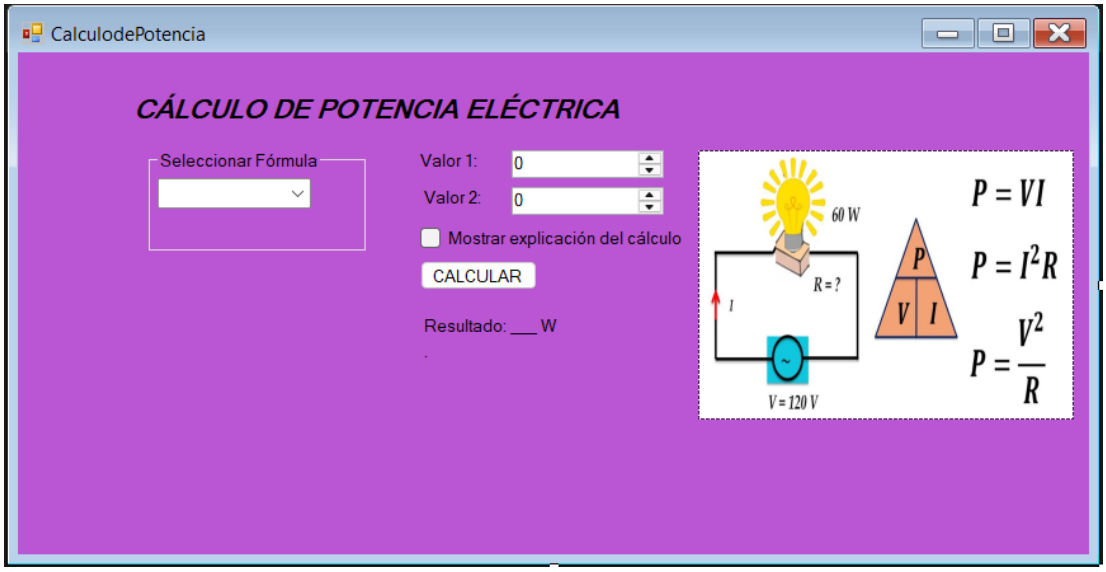


19. **Colocamos los códigos para que funciones los controles que colocamos**

```
#pragma endregion
    private: System::Void ResistenciaParalelo_Load(System::Object^ sender, System::EventArgs
        rb2->Checked = true;
        ActualizarCampos();
    }
    private: System::Void grpSeleccion_Enter(System::Object^ sender, System::EventArgs^ e) {
    }
private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}
private: System::Void rb2_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {
    ActualizarCampos();
}
private: System::Void rb3_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {
    ActualizarCampos();
}
private: System::Void rb4_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {
    ActualizarCampos();
}
private: System::Void textBox2_TextChanged(System::Object^ sender, System::EventArgs^ e) {
}
```

```
361     private: System::Void textBox2_TextChanged(System::Object^ sender, System::EventArgs^ e) {
362     }
363     private: System::Void btnCalcular_Click(System::Object^ sender, System::EventArgs^ e) {
364         if (!Validar(textBox1) || !Validar(textBox2) ||
365             !Validar(textBox3) || !Validar(textBox4))
366             return;
367
368         double sumaInv = 0;
369
370         if (textBox1->Enabled) sumaInv += 1.0 / Convert::ToDouble(textBox1->Text);
371         if (textBox2->Enabled) sumaInv += 1.0 / Convert::ToDouble(textBox2->Text);
372         if (textBox3->Enabled) sumaInv += 1.0 / Convert::ToDouble(textBox3->Text);
373         if (textBox4->Enabled) sumaInv += 1.0 / Convert::ToDouble(textBox4->Text);
374
375         double Req = 1.0 / sumaInv;
376
377         lblResultado->Text = "Resultado: " + Req.ToString("0.00") + " ?";
378     }
379 };
380 }
381
```

20. **De igual manera hacemos con los otros formularios**
    LABEL: nombre del título, nombre del valor y el resultado
    NUMERICUPDOWN: agregar los números
    BUTTON: la opción para calcular
    CHECKBOX: dar una explicación
    PICTUREBOX: poder usar imágenes
    GROUPBOX: crear un marco para agrupar otras funciones
    COMBOBOX: poder escoger que formula usar



```
#pragma endregion
    private: System::Void CalculodePotencia_Load(System::Object^ sender, System::EventArgs^ e) {
    }
    private: System::Void grpFormula_Enter(System::Object^ sender, System::EventArgs^ e) {
    }
private: System::Void cbFormula_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
    if (cbFormula->SelectedIndex == 0) {
        lblDato1->Text = "Voltaje (V):";
        lblDato2->Text = "Corriente (A):";
    }

    else if (cbFormula->SelectedIndex == 1) {
        lblDato1->Text = "Voltaje (V):";
        lblDato2->Text = "Resistencia (Ω):";
    }

    else {
        lblDato1->Text = "Corriente (A):";
        lblDato2->Text = "Resistencia (Ω):";
    }

    lblResultado->Text = "Resultado: ___ W";
    lblExplicacion->Text = "";
}
private: System::Void btnCalcular_Click(System::Object^ sender, System::EventArgs^ e) {
    double a = Convert::ToDouble(numDato1->Value);
    double b = Convert::ToDouble(numDato2->Value);
    double P = 0;

    if (cbFormula->SelectedIndex == 0) {

        P = a * b;
        lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";

        if (chkExplicar->Checked)
            lblExplicacion->Text = "Se usó P = V * I\nP = " + a + " * " + b;
    }

    else if (cbFormula->SelectedIndex == 1) {

        if (b == 0) {
            MessageBox::Show("La resistencia no puede ser 0");
            return;
        }

        P = (a * a) / b;
        lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";

        if (chkExplicar->Checked)
            lblExplicacion->Text = "Se usó P = V^2 / R\nP = " + a + "^2 / " + b;
    }

    else {

        P = (a * a) * b;
        lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";

        if (chkExplicar->Checked)
            lblExplicacion->Text = "Se usó P = I^2 * R\nP = " + a + "^2 * " + b;
    }
}
};
}
```

21. **Por último, la ley de ohm**

    LABEL: poder poner el título, el nombre del valor y el resultado

    NUMERICUPDOWN: poder agregar el numero

    BUTTON: poder calcular

    PICTUREBOX: agregar una imagen

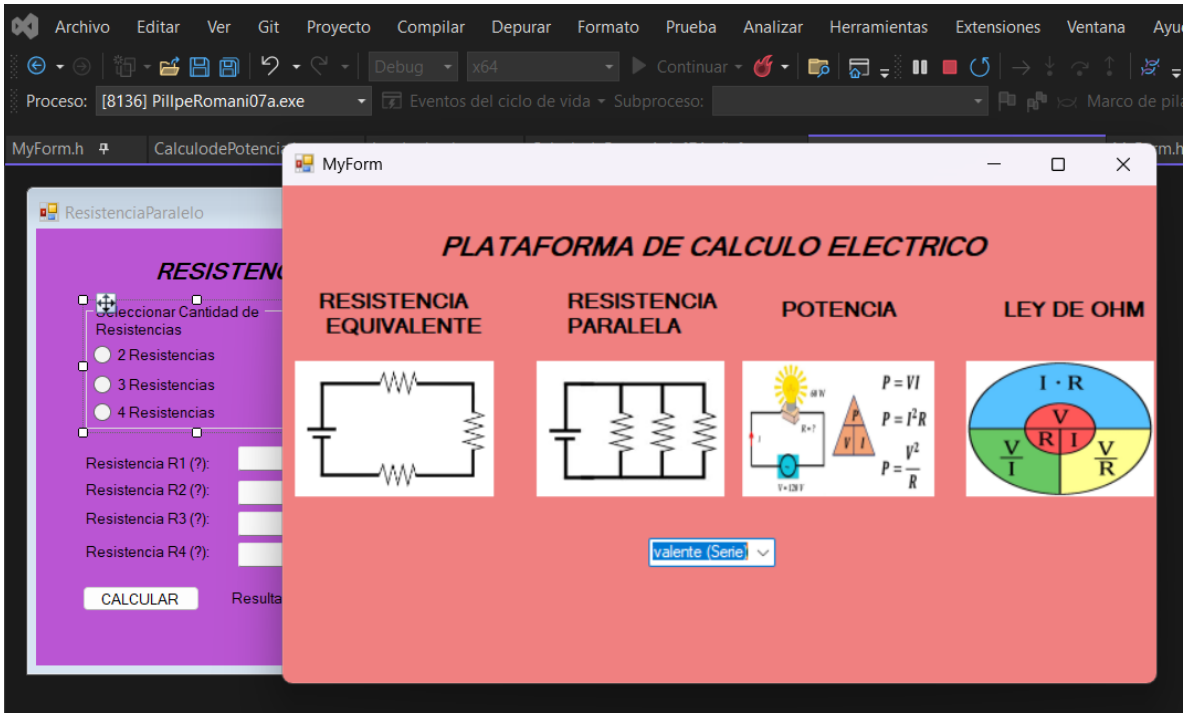
    CHECKBOX: activa la explicación

    COMBOBOX: escoger que ley de ohm usar


```
#pragma endregion
    private: System::Void CalculodePotencia_Load(System::Object^ sender, System::EventArgs^ e) {
    }
    private: System::Void grpFormula_Enter(System::Object^ sender, System::EventArgs^ e) {
    }
private: System::Void cbFormula_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
    if (cbFormula->SelectedIndex == 0) {
        lblDato1->Text = "Voltaje (V):";
        lblDato2->Text = "Corriente (A):";
    }

    else if (cbFormula->SelectedIndex == 1) {
        lblDato1->Text = "Voltaje (V):";
        lblDato2->Text = "Resistencia (Ω):";
    }

    else {
        lblDato1->Text = "Corriente (A):";
        lblDato2->Text = "Resistencia (Ω):";
    }

    lblResultado->Text = "Resultado: ___ W";
    lblExplicacion->Text = "";
}
private: System::Void btnCalcular_Click(System::Object^ sender, System::EventArgs^ e) {
    double a = Convert::ToDouble(numDato1->Value);
    double b = Convert::ToDouble(numDato2->Value);
    double P = 0;

    if (cbFormula->SelectedIndex == 0) {

        P = a * b;
        lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";

        if (chkExplicar->Checked)
            lblExplicacion->Text = "Se usó P = V * I\nP = " + a + " * " + b;
    }

    else if (cbFormula->SelectedIndex == 1) {

        if (b == 0) {
            MessageBox::Show("La resistencia no puede ser 0");
            return;
        }

        P = (a * a) / b;
        lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";

        if (chkExplicar->Checked)
            lblExplicacion->Text = "Se usó P = V^2 / R\nP = " + a + "^2 / " + b;
```

```
282
283         else {
284
285             P = (a * a) * b;
286             lblResultado->Text = "Resultado: " + P.ToString("F2") + " W";
287
288             if (chkExplicar->Checked)
289                 lblExplicacion->Text = "Se usó P = I^2 * R\nP = " + a + "^2 * " + b;
290         }
291     }
292     };
293     }
294
```

22. AL EJECUTARLO QUEDARIA DE LA SIGUINETE MANERA



Instalación y Configuración de MySQL para su Integración con Visual Studio Community 2022

Pillpe Romaní

***Resumen*—En este trabajo se presenta el proceso técnico de instalación y configuración del Sistema Gestor de Bases de Datos (SGBD) MySQL, así como su correcta vinculación con el entorno de desarrollo Visual Studio Community 2022. El objetivo principal es establecer una comunicación exitosa entre una aplicación desarrollada en lenguaje C++/CLI (Windows Forms) y una base de datos local. Se describen los pasos para la descarga de los componentes necesarios, la configuración del servidor y la implementación del código de conexión, validando el proceso mediante un mensaje de confirmación en la interfaz gráfica.**

***Abstract*-- This paper presents the technical process for installing and configuring the MySQL Database Management System (DBMS), as well as its correct integration with the Visual Studio Community 2022 development environment. The main objective is to establish successful communication between an application developed in C++/CLI (Windows Forms) and a local database. The steps for downloading the necessary components, configuring the server, and implementing the connection code are described, validating the process with a confirmation message in the graphical interface.**

I. Introducción

En el desarrollo de software actual, el manejo eficiente de la información constituye un aspecto clave para asegurar el desempeño adecuado de las aplicaciones. Para lograrlo, es esencial emplear sistemas gestores de bases de datos que permitan el almacenamiento, la organización y el tratamiento de los datos de forma confiable y segura.

MySQL es un SGBD relacional de código abierto caracterizado por su alto rendimiento y compatibilidad con diversos lenguajes de programación. La conexión de este sistema con Visual Studio Community 2022 resulta fundamental, ya que permite desarrollar aplicaciones capaces de interactuar directamente con bases de datos en tiempo real, facilitando la implementación de funcionalidades como el registro de usuarios y la gestión de información.

Este informe detalla el procedimiento técnico para lograr dicha integración, sirviendo como apoyo práctico para proyectos académicos y de desarrollo de software.

II. INSTALACIÓN DEL MYSQL

A. descargar MySQL

descargar las herramientas principales:

- **MySQL Installer**

  https://dev.mysql.com/downloads/installer/

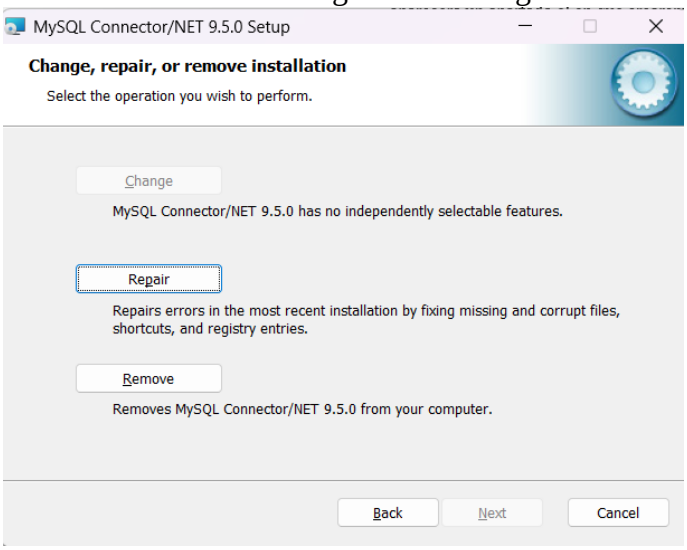
- **MySQL Connector/C++**

  https://downloads.mysql.com/archives/c-cpp/

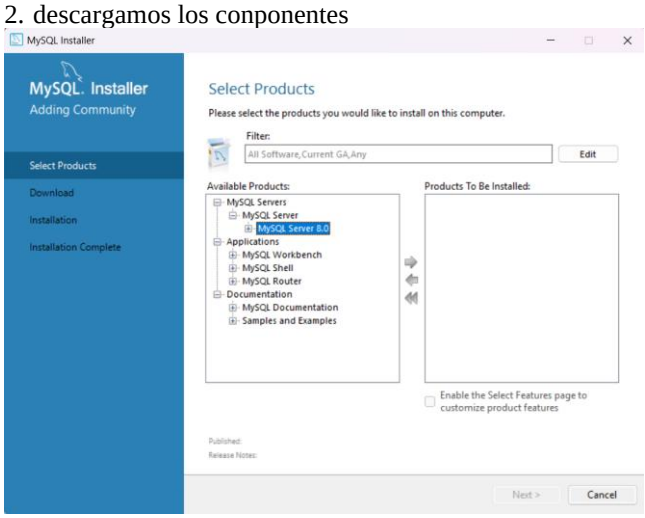
- **MySQL Workbench:**

  https://dev.mysql.com/downloads/workbench/

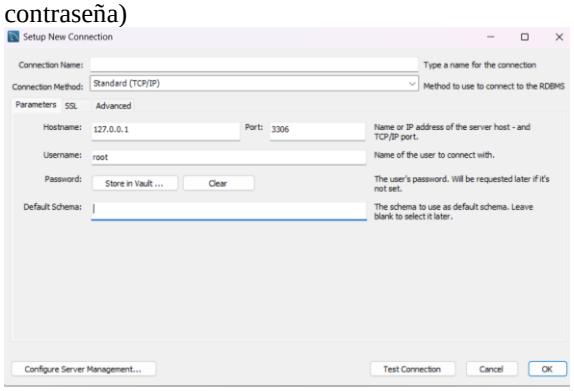
1. descragamos el change



2. descargamos los conponentes

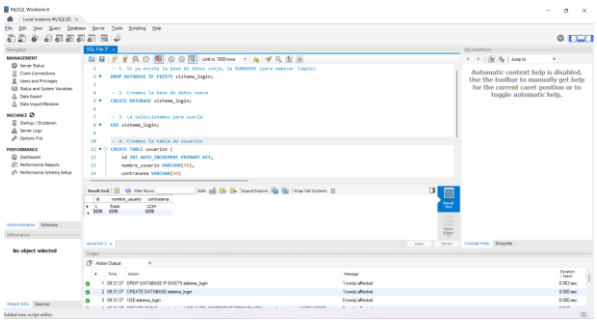


3. creamos la ruta y la contraseña (no olvidar la contraseña)

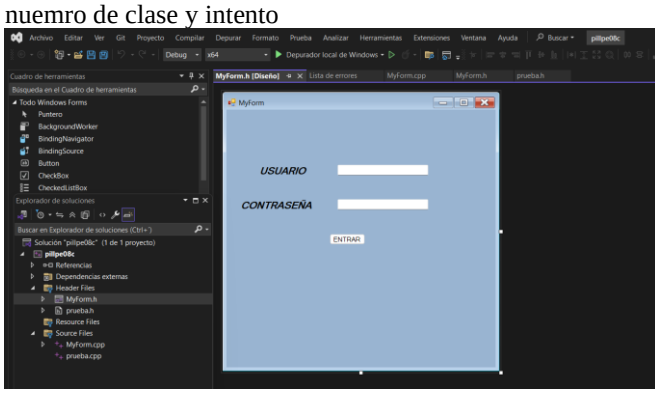


4. Se creo el escritorio de mysql despues de crear la contraseña y el usuario

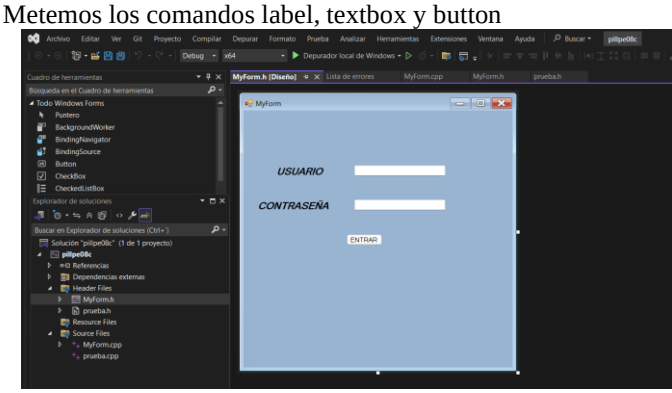
5. Craemos un codigo para que funcio la funcion de usuario y contraseña para vincular co visual studio 2022



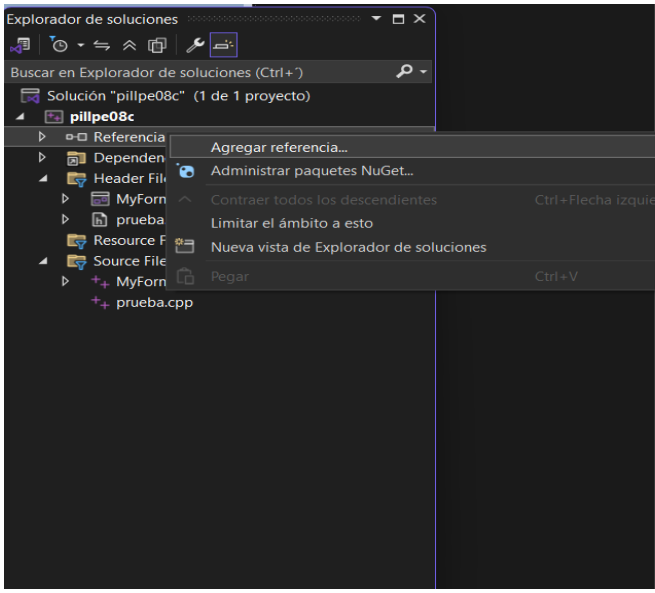
6. Entramos a visual studio 2022 con el apellido el nuemro de clase y intento



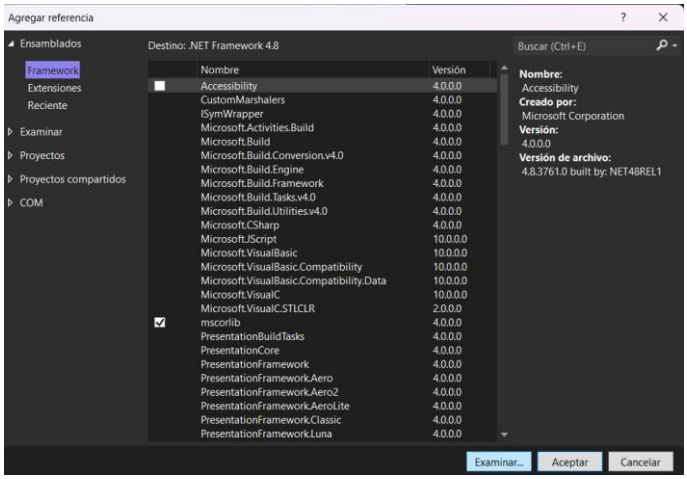
7. Metemos los comandos label, textbox y button



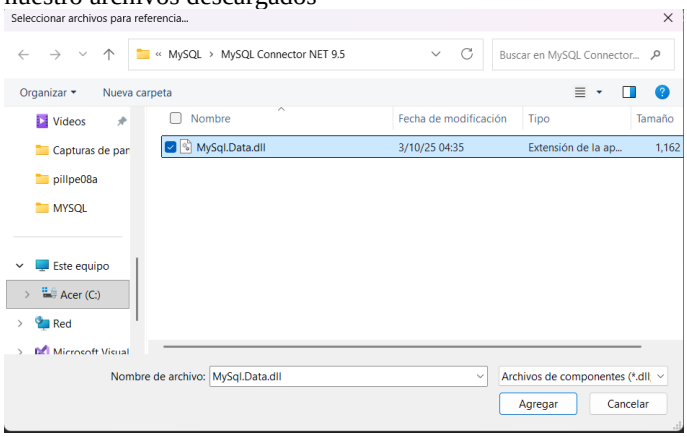
8. Agregar referencia, lo busamos en el explurador de sokuciones debajo del nombre de nuestro proyecto



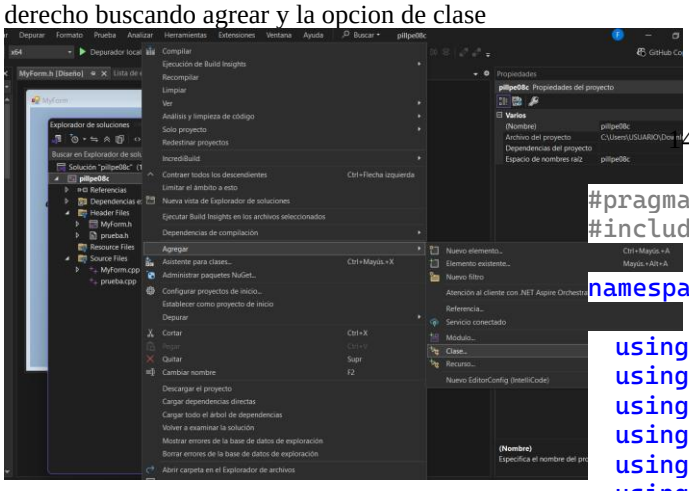
9. Buscamos la opcion framework y en la parte inferior buscamos la opcion de examinar



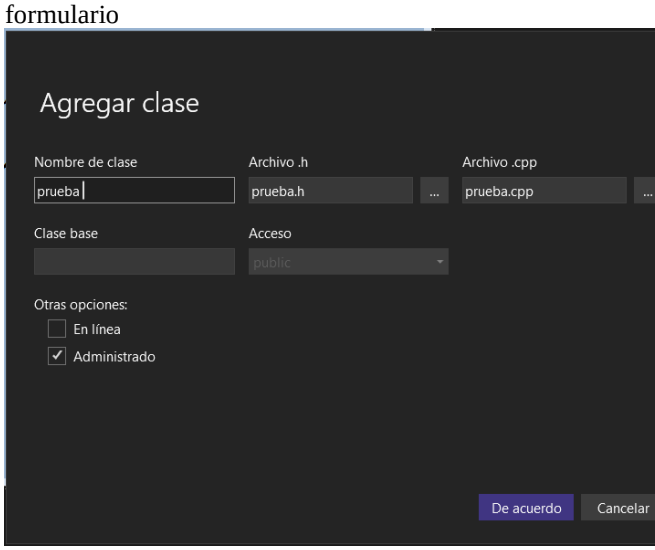
10. Buscamos la opcion de mysql.datta .dll destro de nuestro archivos descargados



11. Creamos una ventana secundaria dando click derecho buscando agrear y la opcion de clase



12. Ponemos el nombre del nueva ventana o formulario



13. Con la ayuda de la ia creamos los codigos en el formulario de prueba.h

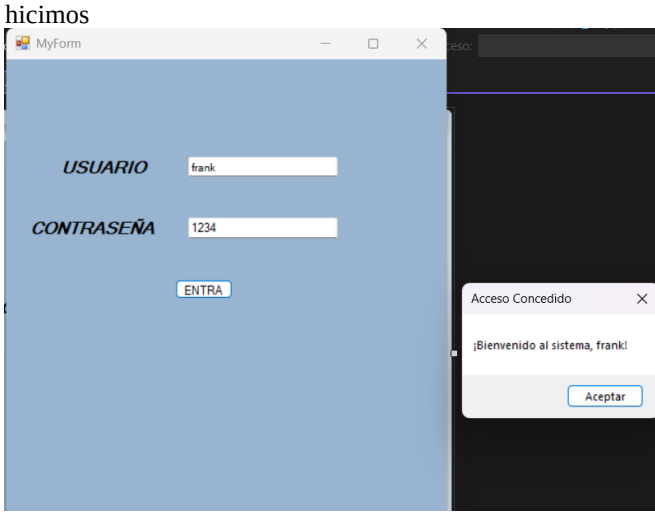
14. Agregamos los nuevos codigos para myform.h

```
#pragma once
#include "Prueba.h"

namespace pillpe08c {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;


  public ref class MyForm : public
System::Windows::Forms::Form
  {
  public:
    MyForm(void)
```

```
    {
      InitializeComponent();

            }

  protected:
    ~MyForm()
    {
      if (components)
      {
        delete components;
      }
    }
  private: System::Windows::Forms::Button^
button1;
  private: System::Windows::Forms::TextBox^
textBox1;
  private: System::Windows::Forms::TextBox^
textBox2;
  private: System::Windows::Forms::Label^
label1;
  private: System::Windows::Forms::Label^
label2;
  protected:

  private:
    System::ComponentModel::Container
^components;

#pragma region Windows Form Designer
generated code
    void InitializeComponent(void)
    {
      this->button1 = (gcnew
System::Windows::Forms::Button());
      this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
      this->textBox2 = (gcnew
System::Windows::Forms::TextBox());
      this->label1 = (gcnew
System::Windows::Forms::Label());
      this->label2 = (gcnew
System::Windows::Forms::Label());
      this->SuspendLayout();
      //
      // button1
      //
      this->button1->Location =
System::Drawing::Point(219, 263);
      this->button1->Name = L"button1";
      this->button1->Size =
System::Drawing::Size(75, 23);
      this->button1->TabIndex = 0;
      this->button1->Text = L"ENTRAR";
      this->button1->UseVisualStyleBackColor =
true;
      this->button1->Click += gcnew
System::EventHandler(this,
&MyForm::button1_Click);
      //
      // textBox1
      //
      this->textBox1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 7.8F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(0)));
      this->textBox1->Location =
System::Drawing::Point(236, 117);
      this->textBox1->Name = L"textBox1";
      this->textBox1->Size =
System::Drawing::Size(192, 22);
      this->textBox1->TabIndex = 1;
      //
      // textBox2
      //
      this->textBox2->Location =
System::Drawing::Point(236, 190);
      this->textBox2->Name = L"textBox2";
      this->textBox2->Size =
System::Drawing::Size(192, 22);
      this->textBox2->TabIndex = 2;
      //
      // label1
      //
      this->label1->AutoSize = true;
      this->label1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 12,
static_cast<System::Drawing::FontStyle>((Syst
em::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
        System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
      this->label1->Location =
System::Drawing::Point(65, 117);
      this->label1->Name = L"label1";
      this->label1->Size =
System::Drawing::Size(109, 50);
      this->label1->TabIndex = 3;
      this->label1->Text = L"USUARIO\r\n\r\n";
      this->label1->Click += gcnew
System::EventHandler(this,
&MyForm::label1_Click);
      //
      // label2
      //
      this->label2->AutoSize = true;
      this->label2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 12,
static_cast<System::Drawing::FontStyle>((Syst
em::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
        System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
      this->label2->Location =
System::Drawing::Point(25, 190);
      this->label2->Name = L"label2";
      this->label2->Size =
System::Drawing::Size(162, 25);
      this->label2->TabIndex = 4;
      this->label2->Text = L"CONTRASEÑA";
```

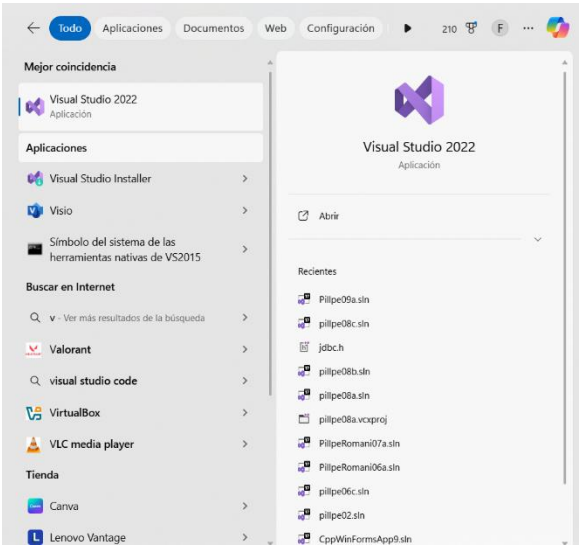
```
      this->label2->Click += gcnew
System::EventHandler(this,
&MyForm::label2_Click);
      //
      // MyForm
      //
      this->AutoScaleDimensions =
System::Drawing::SizeF(8, 16);
      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
      this->BackColor =
System::Drawing::SystemColors::ActiveCaption;
      this->ClientSize =
System::Drawing::Size(571, 546);
      this->Controls->Add(this->label2);
      this->Controls->Add(this->label1);
      this->Controls->Add(this->textBox2);
      this->Controls->Add(this->textBox1);
      this->Controls->Add(this->button1);
      this->Name = L"MyForm";
      this->Text = L"MyForm";
      this->Load += gcnew
System::EventHandler(this,
&MyForm::MyForm_Load);
      this->ResumeLayout(false);
      this->PerformLayout();

    }
#pragma endregion
  private: System::Void
MyForm_Load(System::Object^ sender,
System::EventArgs^ e) {
  }
  private: System::Void
button1_Click(System::Object^ sender,
System::EventArgs^ e) {
    String^ user = textBox1->Text;
    String^ pass = textBox2->Text;

    Prueba::ProbarConexion(user, pass);
  }
  private: System::Void
label1_Click(System::Object^ sender,
System::EventArgs^ e) {
  }
private: System::Void
label2_Click(System::Object^ sender,
System::EventArgs^ e) {
}
};
        }
```

15. Depuramos para poder saber si funciona lo que hicimos

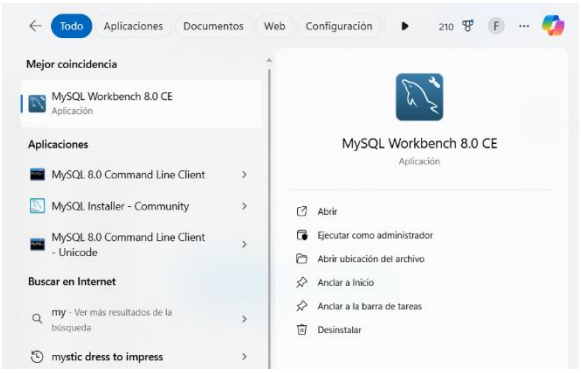


PROBLEMA 1
VERIFICACION DE CONEXION AL SERVIDOR

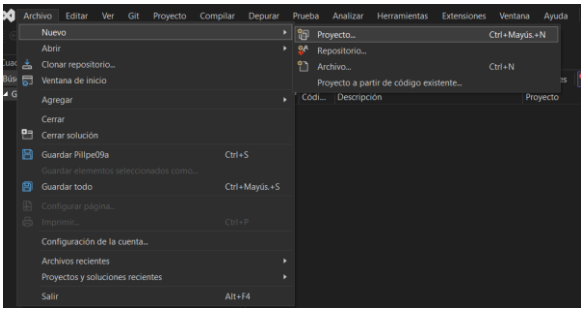
1. Buscamos la aplicación visual studio 2022 en el buscador

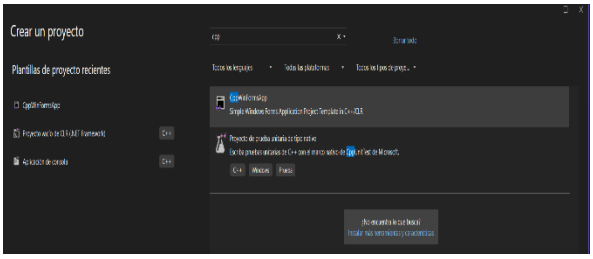


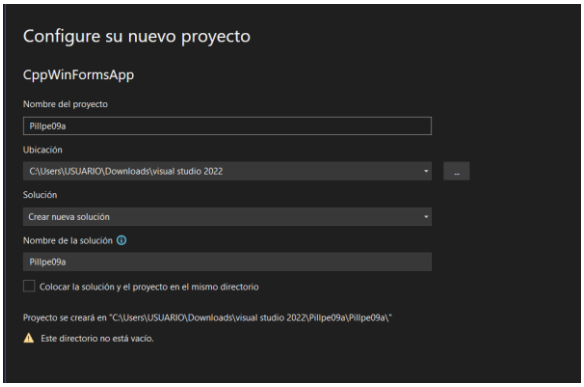
2. También en el mismo buscador buscamos la aplicación mysql



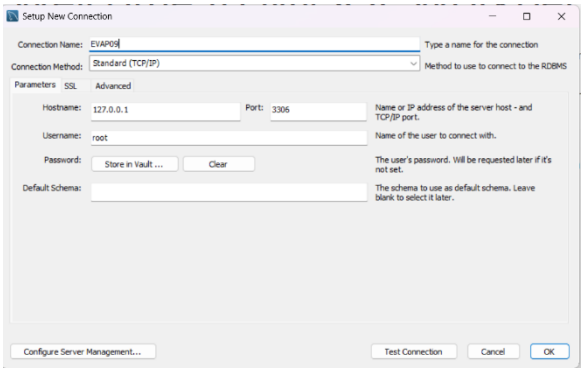
3. Creamos un archivo nuevo con nuestro nombre el numero de la tarea y el abecedario como numero de intento.



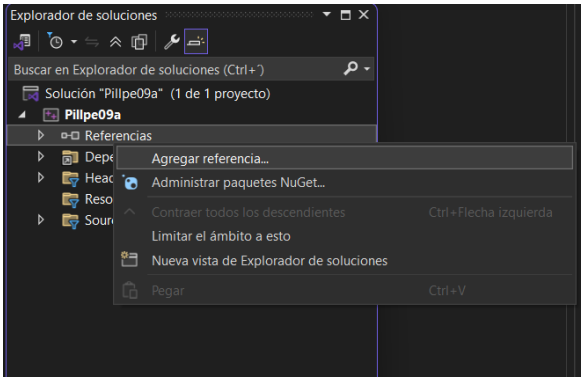


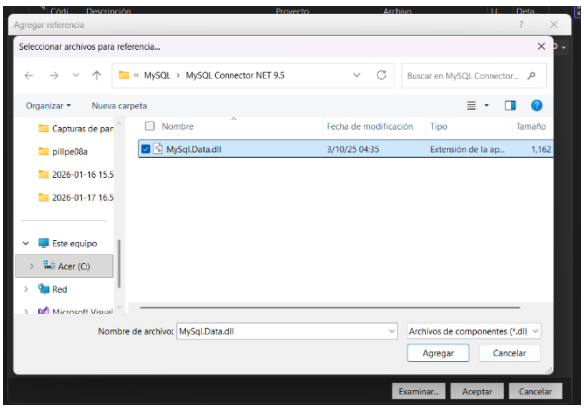


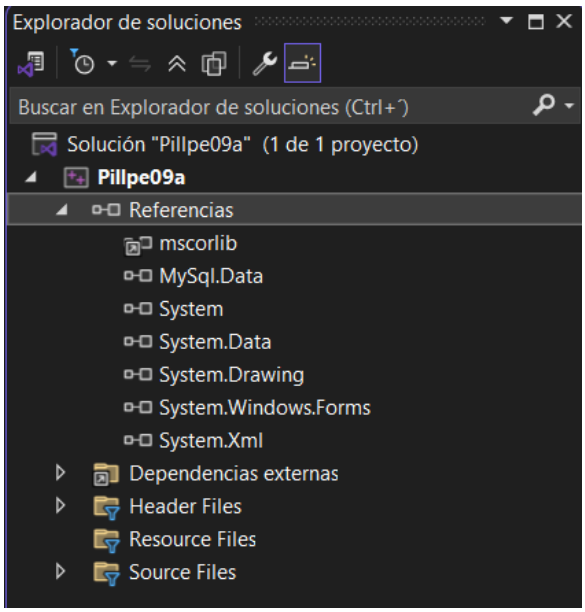
4. Agregamos un nuevo servidor en mysql con el nombre del servidor será el de la semana del trabajo.



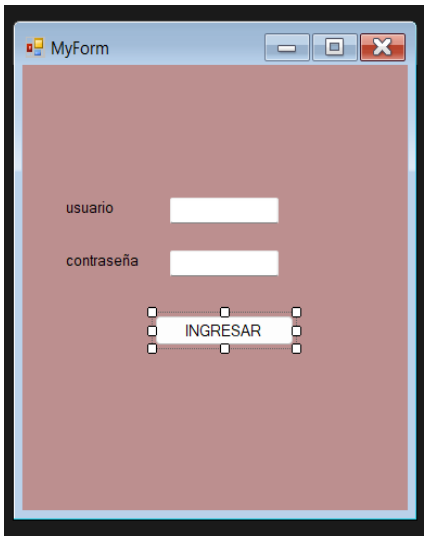
5. Enlazamos nuestro servidor de mysql a visual studio entrando a explorador de soluciones buscamos las referencias y con click derecho vemos la opción de agregar referencia.



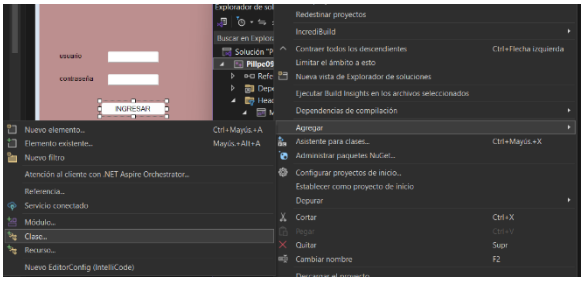




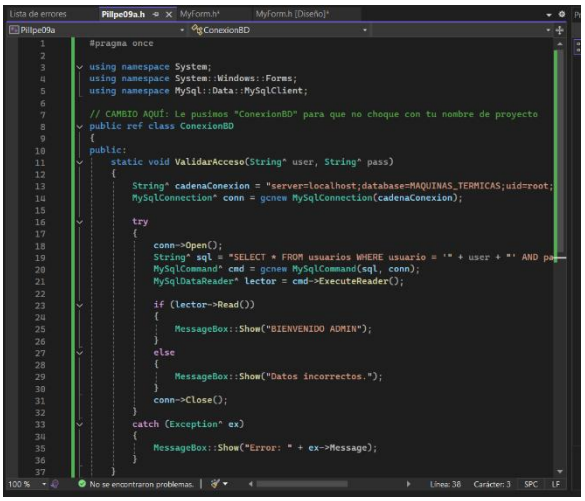
6. Creamos nuestra interfaz gráfica usnado un 2 textbox y un button para poner el usuario y contraseña, el button para ingresar.



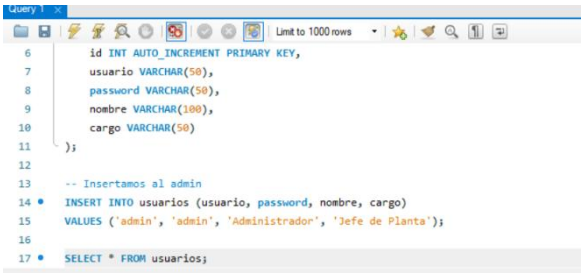
7. Creamos la clase con el nombre del trabajo de la semana que estamos



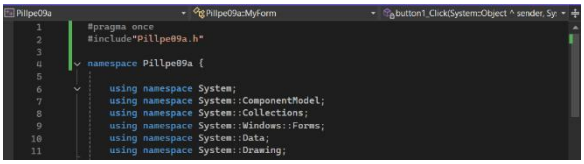
8. Agregamos un código al nuevo arco clase.



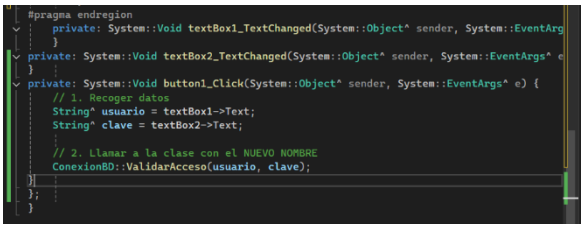
9. Agremamos el código para visual y mysql.



```
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario VARCHAR(50),
    password VARCHAR(50),
    nombre VARCHAR(100),
    cargo VARCHAR(50)
);

-- Insertamos al admin
INSERT INTO usuarios (usuario, password, nombre, cargo)
VALUES ('admin', 'admin', 'Administrador', 'Jefe de Planta');

SELECT * FROM usuarios;
```



```
#pragma once
#include"Pillpe09a.h"

namespace Pillpe09a {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
```



```
#pragma endregion
    private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArg
    }
private: System::Void textBox2_TextChanged(System::Object^ sender, System::EventArgs^ e
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    // 1. Recoger datos
    String^ usuario = textBox1->Text;
    String^ clave = textBox2->Text;

    // 2. Llamar a la clase con el NUEVO NOMBRE
    ConexionBD::ValidarAcceso(usuario, clave);
};
}
```

Código para myform.h

```
#pragma once
#include"Pillpe09a.h"

namespace Pillpe09a {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{

	InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::TextBox^ textBox1;
	protected:
	private: System::Windows::Forms::TextBox^ textBox2;
	private: System::Windows::Forms::Button^ button1;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>

		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->textBox1 = (gcnew System::Windows::Forms::TextBox());
			this->textBox2 = (gcnew System::Windows::Forms::TextBox());
```

```
                    this->button1 =
(gcnew
System::Windows::Forms::Button());
                    this->label1 =
(gcnew
System::Windows::Forms::Label());
                    this->label2 =
(gcnew
System::Windows::Forms::Label());
                    this-
>SuspendLayout();
                    //
                    // textBox1
                    //
                    this->textBox1-
>Location =
System::Drawing::Point(136, 113);
                    this->textBox1-
>Name = L"textBox1";
                    this->textBox1-
>Size = System::Drawing::Size(100,
22);
                    this->textBox1-
>TabIndex = 0;
                    this->textBox1-
>TextChanged += gcnew
System::EventHandler(this,
&MyForm::textBox1_TextChanged);
                    //
                    // textBox2
                    //
                    this->textBox2-
>Location =
System::Drawing::Point(136, 158);
                    this->textBox2-
>Name = L"textBox2";
                    this->textBox2-
>PasswordChar = '*';
                    this->textBox2-
>Size = System::Drawing::Size(100,
22);
                    this->textBox2-
>TabIndex = 1;
                    this->textBox2-
>TextChanged += gcnew
System::EventHandler(this,
&MyForm::textBox2_TextChanged);
                    //
                    // button1
                    //
                    this->button1-
>Location =
System::Drawing::Point(122, 213);
                    this->button1->Name
= L"button1";
                    this->button1->Size
= System::Drawing::Size(128, 26);
                    this->button1-
>TabIndex = 2;
                    this->button1->Text
= L"INGRESAR";
                    this->button1-
>UseVisualStyleBackColor = true;
                    this->button1-
>Click += gcnew
System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // label1
                    //
                    this->label1-
>AutoSize = true;
                    this->label1->Font
= (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

      static_cast<System::Byte>(0)));
                    this->label1-
>Location = System::Drawing::Point(40,
113);
                    this->label1->Name
= L"label1";
                    this->label1->Size
= System::Drawing::Size(71, 20);
                    this->label1-
>TabIndex = 3;
                    this->label1->Text
= L"usuario";
                    //
                    // label2
                    //
                    this->label2-
>AutoSize = true;
                    this->label2->Font
= (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

      static_cast<System::Byte>(0)));
                    this->label2-
>Location = System::Drawing::Point(28,
158);
                    this->label2->Name
= L"label2";
                    this->label2->Size
= System::Drawing::Size(102, 20);
                    this->label2-
>TabIndex = 4;
                    this->label2->Text
= L"contraseña";
```

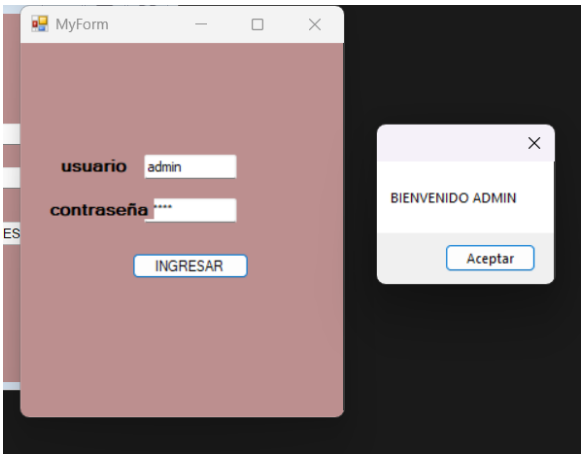
```
                    //
                    // MyForm
                    //
                    this-
>AutoScaleDimensions =
System::Drawing::SizeF(8, 16);
                    this->AutoScaleMode
=
System::Windows::Forms::AutoScaleMode:
:Font;
                    this->BackColor =
System::Drawing::Color::RosyBrown;
                    this->ClientSize =
System::Drawing::Size(355, 379);
                    this->Controls-
>Add(this->label2);
                    this->Controls-
>Add(this->label1);
                    this->Controls-
>Add(this->button1);
                    this->Controls-
>Add(this->textBox2);
                    this->Controls-
>Add(this->textBox1);
                    this->Name =
L"MyForm";
                    this->Text =
L"MyForm";
                    this-
>ResumeLayout(false);
                    this-
>PerformLayout();

            }
#pragma endregion
      private: System::Void
textBox1_TextChanged(System::Object^
sender, System::EventArgs^ e) {
      }
private: System::Void
textBox2_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
button1_Click(System::Object^ sender,
System::EventArgs^ e) {
      // 1. Recoger datos
      String^ usuario = textBox1-
>Text;
      String^ clave = textBox2->Text;

      // 2. Llamar a la clase con el
NUEVO NOMBRE
      ConexionBD::ValidarAcceso(usuari
o, clave);
}
};
}
```

Para la clase Pillpe09a

```
#pragma once

using namespace System;
using namespace
System::Windows::Forms;
using namespace
MySql::Data::MySqlClient;

// CAMBIO AQUÍ: Le pusimos
"ConexionBD" para que no choque con tu
nombre de proyecto
public ref class ConexionBD
{
public:
    static void ValidarAcceso(String^
user, String^ pass)
    {
        String^ cadenaConexion =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadenaConexion);

        try
        {
            conn->Open();
            String^ sql = "SELECT *
FROM usuarios WHERE usuario = '" +
user + "' AND password = '" + pass +
"'";
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            MySqlDataReader^ lector =
cmd->ExecuteReader();

            if (lector->Read())
            {

MessageBox::Show("BIENVENIDO ADMIN");
            }
            else
            {

MessageBox::Show("Datos
incorrectos.");
            }
            conn->Close();
        }
        catch (Exception^ ex)
        {
            MessageBox::Show("Error: "
+ ex->Message);
        }
    }
       };
```

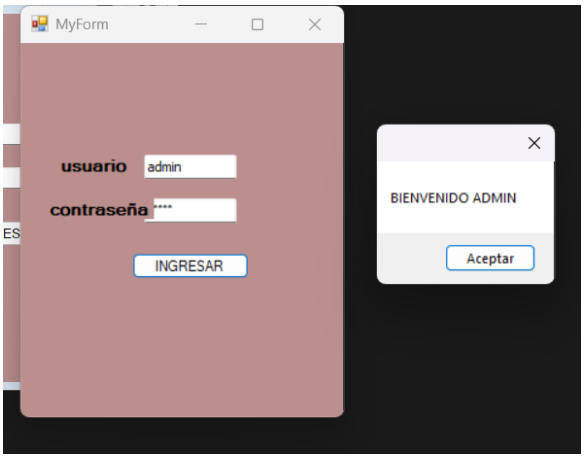
```
Para mysql

DROP DATABASE IF EXISTS
MAQUINAS_TERMICAS;

CREATE DATABASE
MAQUINAS_TERMICAS;

USE MAQUINAS_TERMICAS;


CREATE TABLE usuarios (

   id INT AUTO_INCREMENT PRIMARY
KEY,

   usuario VARCHAR(50),

   password VARCHAR(50),

   nombre VARCHAR(100),

   cargo VARCHAR(50)

);


-- Insertamos al admin

INSERT INTO usuarios (usuario, password,
nombre, cargo)

VALUES ('admin', 'admin', 'Administrador',
'Jefe de Planta');


SELECT * FROM usuarios;
```

10. Se ejecutará en visual studio.

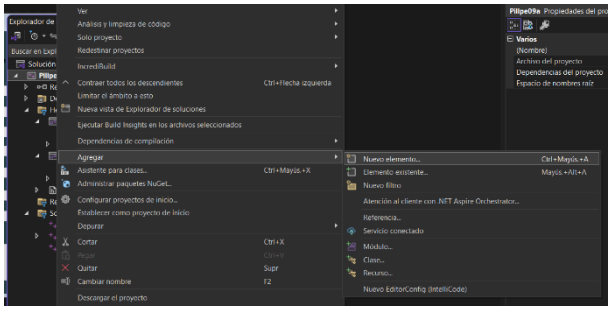


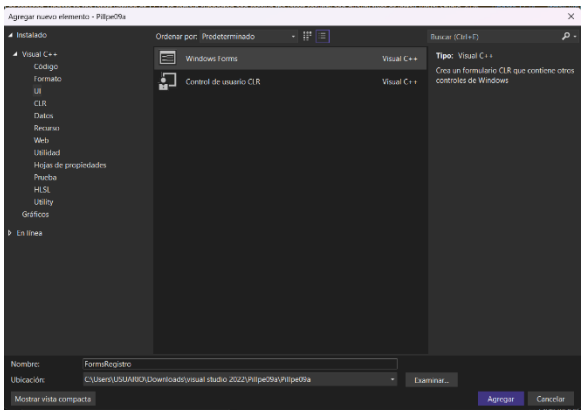
PROBLEMA 2
INSERCIÓN SEGURA DE DATOS

1. Continuamos después de crear nuestro usuario y contraseña

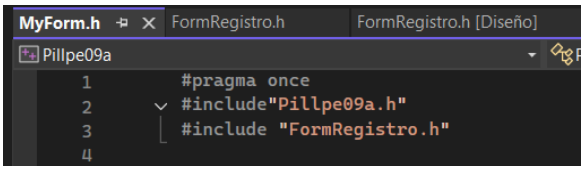


2. Creamos un nuevo elemento dando clic derecho al nombre de nuestro archivo en agregar nuevo elemento y poner nombre a nuestro archivo.

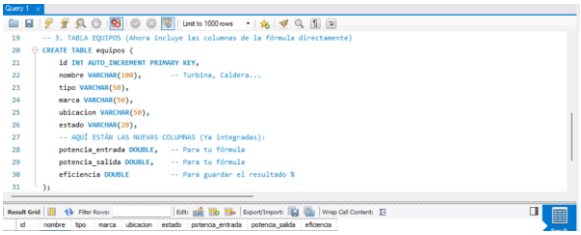




3. Agregamos el #incluide del archivo para poder enlazarlo con nuestro wyform.h de la interfaz grafica



4. Agregamos un código para tener la tabla de máquinas térmicas



CREATE TABLE equipos (

id INT AUTO_INCREMENT PRIMARY KEY,

nombre VARCHAR(100), -- Turbina, Caldera...

tipo VARCHAR(50),

marca VARCHAR(50),

ubicacion VARCHAR(50),

estado VARCHAR(20),

-- AQUÍ ESTÁN LAS NUEVAS COLUMNAS (Ya integradas):

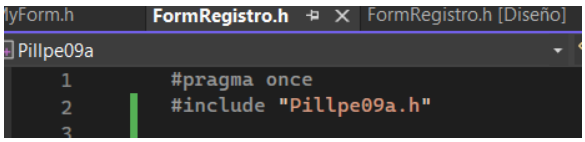
potencia_entrada DOUBLE, -- Para tu fórmula

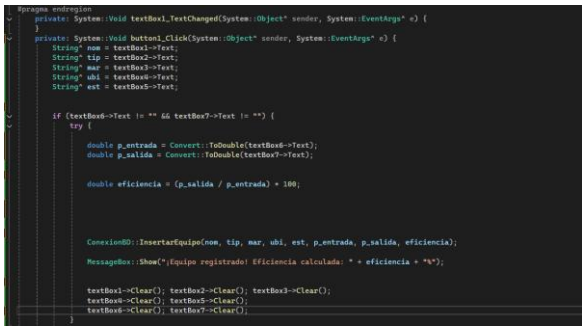
potencia_salida DOUBLE, -- Para tu fórmula

eficiencia DOUBLE -- Para guardar el resultado %

);

5. Código para la nueva ventana de FromRegistro para conecta con la ventana de clase Pillpe09.a.h y crear la interfaz de esta ventana agregando label, textbox y button.





```
#pragma once

#include "Pillpe09a.h"


namespace Pillpe09a {


	using namespace System;

	using namespace
System::ComponentModel;

	using namespace
System::Collections;

	using namespace
System::Windows::Forms;

	using namespace System::Data;

	using namespace System::Drawing;


	/// <summary>

	/// Resumen de FormRegistro

	/// </summary>

	public ref class FormRegistro :
public System::Windows::Forms::Form

	{

	public:

		FormRegistro(void)

		{


	InitializeComponent();

			//

			//TODO: agregar
código de constructor aquí

			//

		}


	protected:

		/// <summary>

		/// Limpiar los recursos que
se estén usando.

		/// </summary>

		~FormRegistro()

		{

			if (components)

			{

				delete
components;

			}

		}
```

```
        private:
System::Windows::Forms::Label^ label1;

        protected:

        private:
System::Windows::Forms::TextBox^
textBox1;

        private:
System::Windows::Forms::Label^ label2;

        private:
System::Windows::Forms::TextBox^
textBox2;

        private:
System::Windows::Forms::Label^ label3;

        private:
System::Windows::Forms::TextBox^
textBox3;

        private:
System::Windows::Forms::Label^ label4;

        private:
System::Windows::Forms::TextBox^
textBox4;

        private:
System::Windows::Forms::Label^ label5;

        private:
System::Windows::Forms::TextBox^
textBox5;

        private:
System::Windows::Forms::Label^ label6;

        private:
System::Windows::Forms::TextBox^
textBox6;

        private:
System::Windows::Forms::Label^ label7;

        private:
System::Windows::Forms::TextBox^
textBox7;

        private:
System::Windows::Forms::Button^ button1;

        private:
System::Windows::Forms::PictureBox^
pictureBox1;

        private:
System::Windows::Forms::Label^ label8;


        private:

                /// <summary>

                /// Variable del diseñador
necesaria.

                /// </summary>


        System::ComponentModel::Contain
er ^components;


#pragma region Windows Form Designer
generated code

                /// <summary>

                /// Método necesario para
admitir el Diseñador. No se puede modificar

                /// el contenido de este
método con el editor de código.

                /// </summary>

                void
InitializeComponent(void)

                {


        System::ComponentModel::Compo
nentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentReso
urceManager(FormRegistro::typeid));

                        this->label1 =
(gcnew System::Windows::Forms::Label());

                        this->textBox1 =
(gcnew
System::Windows::Forms::TextBox());

                        this->label2 =
(gcnew System::Windows::Forms::Label());

                        this->textBox2 =
(gcnew
System::Windows::Forms::TextBox());
```

```
            this->label3 =
(gcnew System::Windows::Forms::Label());
            this->textBox3 =
(gcnew
System::Windows::Forms::TextBox());
            this->label4 =
(gcnew System::Windows::Forms::Label());
            this->textBox4 =
(gcnew
System::Windows::Forms::TextBox());
            this->label5 =
(gcnew System::Windows::Forms::Label());
            this->textBox5 =
(gcnew
System::Windows::Forms::TextBox());
            this->label6 =
(gcnew System::Windows::Forms::Label());
            this->textBox6 =
(gcnew
System::Windows::Forms::TextBox());
            this->label7 =
(gcnew System::Windows::Forms::Label());
            this->textBox7 =
(gcnew
System::Windows::Forms::TextBox());
            this->button1 =
(gcnew
System::Windows::Forms::Button());
            this->pictureBox1
= (gcnew
System::Windows::Forms::PictureBox());
            this->label8 =
(gcnew System::Windows::Forms::Label());


        (cli::safe_cast<System::Component
Model::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
            this-
>SuspendLayout();
            //
            // label1
            //
            this->label1-
>AutoSize = true;
            this->label1-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,


        static_cast<System::Byte>(0)));
            this->label1-
>Location = System::Drawing::Point(75,
114);
            this->label1-
>Name = L"label1";
            this->label1->Size
= System::Drawing::Size(174, 20);
            this->label1-
>TabIndex = 0;
            this->label1-
>Text = L"Nombre del Equipo:";
            this->label1-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label1_Click);
            //
            // textBox1
            //
            this->textBox1-
>Location = System::Drawing::Point(291,
114);
            this->textBox1-
>Name = L"textBox1";
            this->textBox1-
>Size = System::Drawing::Size(100, 22);
            this->textBox1-
>TabIndex = 1;
            this->textBox1-
>TextChanged += gcnew
```

```
System::EventHandler(this,
&FormRegistro::textBox1_TextChanged);
                                //
                                // label2
                                //
                                this->label2-
>AutoSize = true;
                                this->label2-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

        static_cast<System::Byte>(0)));
                                this->label2-
>Location = System::Drawing::Point(75,
158);
                                this->label2-
>Name = L"label2";
                                this->label2->Size
= System::Drawing::Size(153, 20);
                                this->label2-
>TabIndex = 2;
                                this->label2-
>Text = L"Tipo de Máquina:";
                                this->label2-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label2_Click);
                                //
                                // textBox2
                                //
                                this->textBox2-
>Location = System::Drawing::Point(291,
158);
                                this->textBox2-
>Name = L"textBox2";
                                this->textBox2-
>Size = System::Drawing::Size(100, 22);
                                this->textBox2-
>TabIndex = 3;
                                this->textBox2-
>TextChanged += gcnew
System::EventHandler(this,
&FormRegistro::textBox2_TextChanged);
                                //
                                // label3
                                //
                                this->label3-
>AutoSize = true;
                                this->label3-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

        static_cast<System::Byte>(0)));
                                this->label3-
>Location = System::Drawing::Point(75,
200);
                                this->label3-
>Name = L"label3";
                                this->label3->Size
= System::Drawing::Size(145, 20);
                                this->label3-
>TabIndex = 4;
                                this->label3-
>Text = L"Marca / Modelo:";
                                this->label3-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label3_Click);
                                //
                                // textBox3
                                //
                                this->textBox3-
>Location = System::Drawing::Point(291,
200);
```

```
                this->textBox3->Name = L"textBox3";
                this->textBox3->Size = System::Drawing::Size(100, 22);
                this->textBox3->TabIndex = 5;
                this->textBox3->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox3_TextChanged);
                //
                // label4
                //
                this->label4->AutoSize = true;
                this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
            static_cast<System::Byte>(0)));
                this->label4->Location = System::Drawing::Point(75, 240);
                this->label4->Name = L"label4";
                this->label4->Size = System::Drawing::Size(183, 20);
                this->label4->TabIndex = 6;
                this->label4->Text = L"Ubicación en Planta:";
                this->label4->Click += gcnew System::EventHandler(this, &FormRegistro::label4_Click);
                //
                // textBox4
                //
                this->textBox4->Location = System::Drawing::Point(291, 240);
                this->textBox4->Name = L"textBox4";
                this->textBox4->Size = System::Drawing::Size(100, 22);
                this->textBox4->TabIndex = 7;
                this->textBox4->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox4_TextChanged);
                //
                // label5
                //
                this->label5->AutoSize = true;
                this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
            static_cast<System::Byte>(0)));
                this->label5->Location = System::Drawing::Point(75, 285);
                this->label5->Name = L"label5";
                this->label5->Size = System::Drawing::Size(132, 20);
                this->label5->TabIndex = 8;
                this->label5->Text = L"Estado Actual:";
                this->label5->Click += gcnew
```

```
System::EventHandler(this,
&FormRegistro::label5_Click);

                    //

                    // textBox5

                    //

                    this->textBox5-
>Location = System::Drawing::Point(291,
285);

                    this->textBox5-
>Name = L"textBox5";

                    this->textBox5-
>Size = System::Drawing::Size(100, 22);

                    this->textBox5-
>TabIndex = 9;

                    this->textBox5-
>TextChanged += gcnew
System::EventHandler(this,
&FormRegistro::textBox5_TextChanged);

                    //

                    // label6

                    //

                    this->label6-
>AutoSize = true;

                    this->label6-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,


        static_cast<System::Byte>(0)));

                    this->label6-
>Location = System::Drawing::Point(75,
335);

                    this->label6-
>Name = L"label6";

                    this->label6->Size
= System::Drawing::Size(205, 20);

                    this->label6-
>TabIndex = 10;

                    this->label6-
>Text = L"Potencia Entrada (kW):";

                    this->label6-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label6_Click);

                    //

                    // textBox6

                    //

                    this->textBox6-
>Location = System::Drawing::Point(291,
333);

                    this->textBox6-
>Name = L"textBox6";

                    this->textBox6-
>Size = System::Drawing::Size(100, 22);

                    this->textBox6-
>TabIndex = 11;

                    this->textBox6-
>TextChanged += gcnew
System::EventHandler(this,
&FormRegistro::textBox6_TextChanged);

                    //

                    // label7

                    //

                    this->label7-
>AutoSize = true;

                    this->label7-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,


        static_cast<System::Byte>(0)));

                    this->label7-
>Location = System::Drawing::Point(75,
376);

                    this->label7-
>Name = L"label7";
```

```
                this->label7->Size
= System::Drawing::Size(192, 20);

                this->label7-
>TabIndex = 12;

                this->label7-
>Text = L"Potencia Salida (kW):";

                this->label7-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label7_Click);

                //

                // textBox7

                //

                this->textBox7-
>Location = System::Drawing::Point(291,
376);

                this->textBox7-
>Name = L"textBox7";

                this->textBox7-
>Size = System::Drawing::Size(100, 22);

                this->textBox7-
>TabIndex = 13;

                this->textBox7-
>TextChanged += gcnew
System::EventHandler(this,
&FormRegistro::textBox7_TextChanged);

                //

                // button1

                //

                this->button1-
>Font = (gcnew
System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,


        static_cast<System::Byte>(0)));

                this->button1-
>Location = System::Drawing::Point(109,
444);

                this->button1-
>Name = L"button1";

                this->button1-
>Size = System::Drawing::Size(282, 79);

                this->button1-
>TabIndex = 14;

                this->button1-
>Text = L"REGISTRAR Y CALCULAR";

                this->button1-
>UseVisualStyleBackColor = true;

                this->button1-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::button1_Click);

                //

                // pictureBox1

                //

                this-
>pictureBox1->Image =
(cli::safe_cast<System::Drawing::Image^>(r
esources-
>GetObject(L"pictureBox1.Image")));

                this-
>pictureBox1->Location =
System::Drawing::Point(450, 116);

                this-
>pictureBox1->Name = L"pictureBox1";

                this-
>pictureBox1->Size =
System::Drawing::Size(310, 282);

                this-
>pictureBox1->SizeMode =
System::Windows::Forms::PictureBoxSize
Mode::StretchImage;

                this-
>pictureBox1->TabIndex = 15;

                this-
>pictureBox1->TabStop = false;

                //

                // label8
```

```
            //
            this->label8->AutoSize = true;
            this->label8->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(0)));
            this->label8->Location = System::Drawing::Point(170, 50);
            this->label8->Name = L"label8";
            this->label8->Size = System::Drawing::Size(494, 20);
            this->label8->TabIndex = 16;
            this->label8->Text = L"Gestión y Eficiencia de Máquinas Térmicas e Industriales";
            this->label8->Click += gcnew System::EventHandler(this, &FormRegistro::label8_Click);
            //
            // FormRegistro
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->BackColor = System::Drawing::Color::DarkSlateGray;
            this->ClientSize = System::Drawing::Size(842, 549);
            this->Controls->Add(this->label8);
            this->Controls->Add(this->pictureBox1);
            this->Controls->Add(this->button1);
            this->Controls->Add(this->textBox7);
            this->Controls->Add(this->label7);
            this->Controls->Add(this->textBox6);
            this->Controls->Add(this->label6);
            this->Controls->Add(this->textBox5);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->textBox4);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->textBox3);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->textBox2);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->textBox1);
            this->Controls->Add(this->label1);
            this->Name = L"FormRegistro";
            this->Text = L"FormRegistro";
```

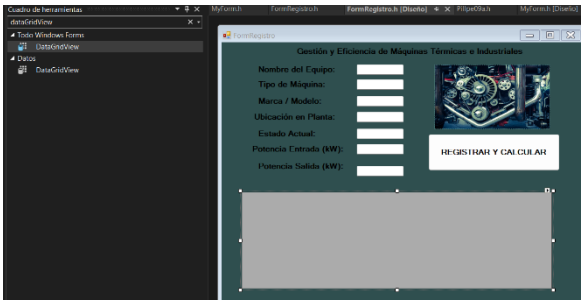
```
            this->Load +=
gcnew System::EventHandler(this,
&FormRegistro::FormRegistro_Load);


        (cli::safe_cast<System::Component
Model::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
            this-
>ResumeLayout(false);
            this-
>PerformLayout();


        }
#pragma endregion
        private: System::Void
textBox1_TextChanged(System::Object^
sender, System::EventArgs^ e) {
        }
        private: System::Void
button1_Click(System::Object^ sender,
System::EventArgs^ e) {
            String^ nom = textBox1-
>Text;
            String^ tip = textBox2-
>Text;
            String^ mar = textBox3-
>Text;
            String^ ubi = textBox4-
>Text;
            String^ est = textBox5-
>Text;



            if (textBox6->Text != ""
&& textBox7->Text != "") {
                try {
                    double
p_entrada = Convert::ToDouble(textBox6-
>Text);
                    double
p_salida = Convert::ToDouble(textBox7-
>Text);



                    double
eficiencia = (p_salida / p_entrada) * 100;



        ConexionBD::InsertarEquipo(nom,
tip, mar, ubi, est, p_entrada, p_salida,
eficiencia);



        MessageBox::Show("¡Equipo
registrado! Eficiencia calculada: " +
eficiencia + "%");



        textBox1->Clear(); textBox2-
>Clear(); textBox3->Clear();

        textBox4->Clear(); textBox5-
>Clear();

        textBox6->Clear(); textBox7-
>Clear();
                }
                catch (...) {

        MessageBox::Show("Error: Por
```

```
favor ingresa solo números en las
potencias.");
                        }
                }
                else {

        MessageBox::Show("Por favor
llena todos los datos, incluyendo
potencias.");
                }
        }
private: System::Void
textBox2_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
textBox3_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
textBox4_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
textBox5_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
textBox6_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
textBox7_TextChanged(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
label1_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label2_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label3_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label4_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
FormRegistro_Load(System::Object^
sender, System::EventArgs^ e) {
}
private: System::Void
label5_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label6_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label7_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void
label8_Click(System::Object^ sender,
System::EventArgs^ e) {
}
};
}
```

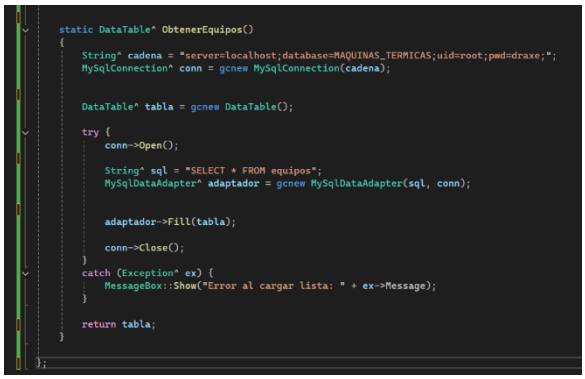
6. Depuramos y vemos nuestro resultado.

PROBLEMA 3
VISUALIZACION DE DATOS EN CUADRICULA

1. Entramos a cuadro de herramientas y buscamos dataGridView para ver los datos en cuadricula



2. En la ventana Pillpe09a agregamos el siguiente código esto iría al final

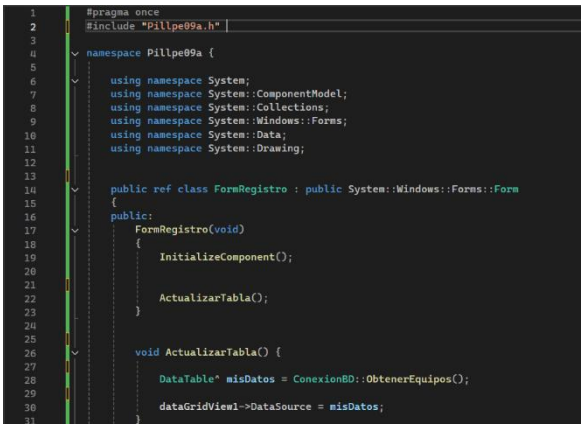


```
static DataTable^ ObtenerEquipos()
{
    String^ cadena = "server=localhost;database=MAQUINAS_TERMICAS;uid=root;pwd=draxe;";
    MySqlConnection^ conn = gcnew MySqlConnection(cadena);

    DataTable^ tabla = gcnew DataTable();

    try {
        conn->Open();

        String^ sql = "SELECT * FROM equipos";
        MySqlDataAdapter^ adaptador = gcnew MySqlDataAdapter(sql, conn);

        adaptador->Fill(tabla);

        conn->Close();
    }
    catch (Exception^ ex) {
        MessageBox::Show("Error al cargar lista: " + ex->Message);
    }

    return tabla;
}
};
```

```
#pragma once

using namespace System;
using namespace
System::Windows::Forms;
using namespace
MySql::Data::MySqlClient;
using namespace System::Data;

public ref class ConexionBD
{
public:

    static bool ValidarAcceso(String^
user, String^ pass)
    {
        String^ cadenaConexion =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadenaConexion);
        try
        {
            conn->Open();
            String^ sql = "SELECT *
FROM usuarios WHERE usuario = '" +
user + "' AND password = '" + pass +
"'";
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            MySqlDataReader^ lector =
cmd->ExecuteReader();

            if (lector->Read())
            {
                conn->Close();
                return true;
            }
            else
            {
                conn->Close();
                return false;
            }
        }
        catch (Exception^ ex)
        {
            MessageBox::Show("Error de
conexión: " + ex->Message);
            return false;
        }
    }


    static void InsertarEquipo(String^
nom, String^ tip, String^ mar, String^
ubi, String^ est, double p_ent, double
p_sal, double efic)
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);

        try {
            conn->Open();
            String^ sql = "INSERT INTO
equipos (nombre, tipo, marca,
ubicacion, estado, potencia_entrada,
potencia_salida, eficiencia) VALUES
('" + nom + "', '" + tip + "', '" +
mar + "', '" + ubi + "', '" + est +
"', " + p_ent + ", " + p_sal + ", " +
efic + ")";
```

```
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            cmd->ExecuteNonQuery();
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
guardar en BD: " + ex->Message);
        }
    }


    static DataTable^ ObtenerEquipos()
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);


        DataTable^ tabla = gcnew
DataTable();

        try {
            conn->Open();

            String^ sql = "SELECT *
FROM equipos";
            MySqlDataAdapter^
adaptador = gcnew
MySqlDataAdapter(sql, conn);


            adaptador->Fill(tabla);

            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
cargar lista: " + ex->Message);
        }

        return tabla;
    }

      };
```

3. Para formRegistro es el siguiente código.



```
#pragma once
#include "Pillpe09a.h"

namespace Pillpe09a {

        using namespace System;
        using namespace
System::ComponentModel;
        using namespace
System::Collections;
        using namespace
System::Windows::Forms;
        using namespace System::Data;
        using namespace
System::Drawing;


        public ref class FormRegistro :
public System::Windows::Forms::Form
        {
        public:
                FormRegistro(void)
                {

        InitializeComponent();



        ActualizarTabla();
                }


                void ActualizarTabla() {
```

```
                DataTable^
misDatos =
ConexionBD::ObtenerEquipos();

                dataGridView1-
>DataSource = misDatos;
            }

        protected:
            /// <summary>
            /// Limpiar los recursos
que se estén usando.
            /// </summary>
            ~FormRegistro()
            {
                if (components)
                {
                    delete
components;
                }
            }
        private:
System::Windows::Forms::Label^
label1;
        protected:
        private:
System::Windows::Forms::TextBox^
textBox1;
        private:
System::Windows::Forms::Label^
label2;
        private:
System::Windows::Forms::TextBox^
textBox2;
        private:
System::Windows::Forms::Label^
label3;
        private:
System::Windows::Forms::TextBox^
textBox3;
        private:
System::Windows::Forms::Label^
label4;
        private:
System::Windows::Forms::TextBox^
textBox4;
        private:
System::Windows::Forms::Label^
label5;
        private:
System::Windows::Forms::TextBox^
textBox5;
        private:
System::Windows::Forms::Label^
label6;
        private:
System::Windows::Forms::TextBox^
textBox6;
        private:
System::Windows::Forms::Label^
label7;
        private:
System::Windows::Forms::TextBox^
textBox7;
        private:
System::Windows::Forms::Button^
button1;

        private:
System::Windows::Forms::Label^
label8;
        private:
System::Windows::Forms::DataGridVie
w^ dataGridView1;
        private:
System::Windows::Forms::PictureBox^
pictureBox1;

        private:
            /// <summary>
            /// Variable del
diseñador necesaria.
            /// </summary>

        System::ComponentModel::Con
tainer^ components;

#pragma region Windows Form
Designer generated code
            /// <summary>
            /// Método necesario
para admitir el Diseñador. No se puede
modificar
            /// el contenido de este
método con el editor de código.
            /// </summary>
            void
InitializeComponent(void)
            {

        System::ComponentModel::Co
```

```
mponentResourceManager^ resources =
(gcnew
System::ComponentModel::Component
ResourceManager(FormRegistro::typeid
));
                this->label1 =
(gcnew
System::Windows::Forms::Label());
                this->textBox1
= (gcnew
System::Windows::Forms::TextBox());
                this->label2 =
(gcnew
System::Windows::Forms::Label());
                this->textBox2
= (gcnew
System::Windows::Forms::TextBox());
                this->label3 =
(gcnew
System::Windows::Forms::Label());
                this->textBox3
= (gcnew
System::Windows::Forms::TextBox());
                this->label4 =
(gcnew
System::Windows::Forms::Label());
                this->textBox4
= (gcnew
System::Windows::Forms::TextBox());
                this->label5 =
(gcnew
System::Windows::Forms::Label());
                this->textBox5
= (gcnew
System::Windows::Forms::TextBox());
                this->label6 =
(gcnew
System::Windows::Forms::Label());
                this->textBox6
= (gcnew
System::Windows::Forms::TextBox());
                this->label7 =
(gcnew
System::Windows::Forms::Label());
                this->textBox7
= (gcnew
System::Windows::Forms::TextBox());
                this->button1 =
(gcnew
System::Windows::Forms::Button());
                this->label8 =
(gcnew
System::Windows::Forms::Label());
                this-
>dataGridView1 = (gcnew
System::Windows::Forms::DataGridVie
w());
                this-
>pictureBox1 = (gcnew
System::Windows::Forms::PictureBox()
);

        (cli::safe_cast<System::Compon
entModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();

        (cli::safe_cast<System::Compon
entModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
                this-
>SuspendLayout();
                //
                // label1
                //
                this->label1-
>AutoSize = true;
                this->label1-
>Font = (gcnew
System::Drawing::Font(L"Microsoft
Sans Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

        static_cast<System::Byte>(0)));
                this->label1-
>Location =
System::Drawing::Point(75, 48);
                this->label1-
>Name = L"label1";
                this->label1-
>Size = System::Drawing::Size(174,
20);
                this->label1-
>TabIndex = 0;
                this->label1-
>Text = L"Nombre del Equipo:";
                this->label1-
>Click += gcnew
System::EventHandler(this,
&FormRegistro::label1_Click);
                //
                // textBox1
```

```
                        //
                        this->textBox1->Location = System::Drawing::Point(291, 48);
                        this->textBox1->Name = L"textBox1";
                        this->textBox1->Size = System::Drawing::Size(100, 22);
                        this->textBox1->TabIndex = 1;
                        this->textBox1->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox1_TextChanged);
                        //
                        // label2
                        //
                        this->label2->AutoSize = true;
                        this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                        this->label2->Location = System::Drawing::Point(75, 80);
                        this->label2->Name = L"label2";
                        this->label2->Size = System::Drawing::Size(153, 20);
                        this->label2->TabIndex = 2;
                        this->label2->Text = L"Tipo de Máquina:";
                        this->label2->Click += gcnew System::EventHandler(this, &FormRegistro::label2_Click);
                        //
                        // textBox2
                        //
                        this->textBox2->Location = System::Drawing::Point(291, 80);
                        this->textBox2->Name = L"textBox2";
                        this->textBox2->Size = System::Drawing::Size(100, 22);
                        this->textBox2->TabIndex = 3;
                        this->textBox2->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox2_TextChanged);
                        //
                        // label3
                        //
                        this->label3->AutoSize = true;
                        this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                        this->label3->Location = System::Drawing::Point(75, 116);
                        this->label3->Name = L"label3";
                        this->label3->Size = System::Drawing::Size(145, 20);
                        this->label3->TabIndex = 4;
                        this->label3->Text = L"Marca / Modelo:";
                        this->label3->Click += gcnew System::EventHandler(this, &FormRegistro::label3_Click);
                        //
                        // textBox3
                        //
                        this->textBox3->Location = System::Drawing::Point(291, 116);
                        this->textBox3->Name = L"textBox3";
                        this->textBox3->Size = System::Drawing::Size(100,
```

```
22);
                this->textBox3->TabIndex = 5;
                this->textBox3->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox3_TextChanged);
                //
                // label4
                //
                this->label4->AutoSize = true;
                this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                    static_cast<System::Byte>(0)));
                this->label4->Location = System::Drawing::Point(66, 150);
                this->label4->Name = L"label4";
                this->label4->Size = System::Drawing::Size(183, 20);
                this->label4->TabIndex = 6;
                this->label4->Text = L"Ubicación en Planta:";
                this->label4->Click += gcnew System::EventHandler(this, &FormRegistro::label4_Click);
                //
                // textBox4
                //
                this->textBox4->Location = System::Drawing::Point(291, 150);
                this->textBox4->Name = L"textBox4";
                this->textBox4->Size = System::Drawing::Size(100, 22);
                this->textBox4->TabIndex = 7;
                this->textBox4->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox4_TextChanged);
                //
                // label5
                //
                this->label5->AutoSize = true;
                this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                    static_cast<System::Byte>(0)));
                this->label5->Location = System::Drawing::Point(75, 186);
                this->label5->Name = L"label5";
                this->label5->Size = System::Drawing::Size(132, 20);
                this->label5->TabIndex = 8;
                this->label5->Text = L"Estado Actual:";
                this->label5->Click += gcnew System::EventHandler(this, &FormRegistro::label5_Click);
                //
                // textBox5
                //
                this->textBox5->Location = System::Drawing::Point(291, 186);
                this->textBox5->Name = L"textBox5";
                this->textBox5->Size = System::Drawing::Size(100, 22);
                this->textBox5->TabIndex = 9;
                this->textBox5->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox5_TextChanged);
```

```
                //
                // label6
                //
                this->label6->AutoSize = true;
                this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                    static_cast<System::Byte>(0)));
                this->label6->Location = System::Drawing::Point(62, 219);
                this->label6->Name = L"label6";
                this->label6->Size = System::Drawing::Size(205, 20);
                this->label6->TabIndex = 10;
                this->label6->Text = L"Potencia Entrada (kW):";
                this->label6->Click += gcnew System::EventHandler(this, &FormRegistro::label6_Click);
                //
                // textBox6
                //
                this->textBox6->Location = System::Drawing::Point(291, 219);
                this->textBox6->Name = L"textBox6";
                this->textBox6->Size = System::Drawing::Size(100, 22);
                this->textBox6->TabIndex = 11;
                this->textBox6->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox6_TextChanged);
                //
                // label7
                //
                this->label7->AutoSize = true;
                this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                    static_cast<System::Byte>(0)));
                this->label7->Location = System::Drawing::Point(75, 256);
                this->label7->Name = L"label7";
                this->label7->Size = System::Drawing::Size(192, 20);
                this->label7->TabIndex = 12;
                this->label7->Text = L"Potencia Salida (kW):";
                this->label7->Click += gcnew System::EventHandler(this, &FormRegistro::label7_Click);
                //
                // textBox7
                //
                this->textBox7->Location = System::Drawing::Point(291, 266);
                this->textBox7->Name = L"textBox7";
                this->textBox7->Size = System::Drawing::Size(100, 22);
                this->textBox7->TabIndex = 13;
                this->textBox7->TextChanged += gcnew System::EventHandler(this, &FormRegistro::textBox7_TextChanged);
                //
                // button1
                //
                this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Bold,
```

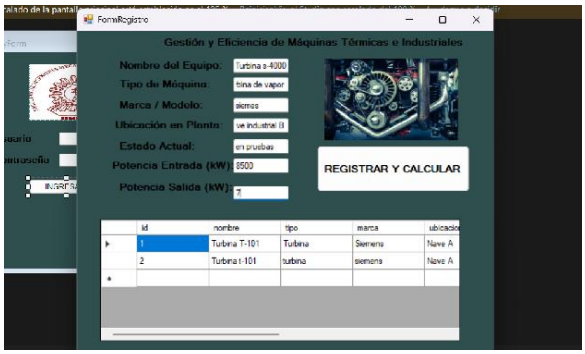
```
System::Drawing::GraphicsUnit::Point,

        static_cast<System::Byte>(0)));
                this->button1->Location =
System::Drawing::Point(445, 197);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(282,
79);
                this->button1->TabIndex = 14;
                this->button1->Text = L"REGISTRAR Y
CALCULAR";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew
System::EventHandler(this,
&FormRegistro::button1_Click);
                //
                // label8
                //
                this->label8->AutoSize = true;
                this->label8->Font = (gcnew
System::Drawing::Font(L"Microsoft
Sans Serif", 10.2F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

        static_cast<System::Byte>(0)));
                this->label8->Location =
System::Drawing::Point(158, 9);
                this->label8->Name = L"label8";
                this->label8->Size = System::Drawing::Size(494,
20);
                this->label8->TabIndex = 16;
                this->label8->Text = L"Gestión y Eficiencia de
Máquinas Térmicas e Industriales";
                this->label8->Click += gcnew
System::EventHandler(this,
&FormRegistro::label8_Click);
                //
                //
dataGridView1
                //
                this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridVie
wColumnHeadersHeightSizeMode::Aut
oSize;
                this->dataGridView1->Location =
System::Drawing::Point(43, 322);
                this->dataGridView1->Name =
L"dataGridView1";
                this->dataGridView1->RowHeadersWidth =
51;
                this->dataGridView1->RowTemplate->Height = 24;
                this->dataGridView1->Size =
System::Drawing::Size(668, 208);
                this->dataGridView1->TabIndex = 17;
                //
                // pictureBox1
                //
                this->pictureBox1->Image =
(cli::safe_cast<System::Drawing::Image
^>(resources->GetObject(L"pictureBox1.Image")));
                this->pictureBox1->Location =
System::Drawing::Point(458, 48);
                this->pictureBox1->Name =
L"pictureBox1";
                this->pictureBox1->Size =
System::Drawing::Size(248, 139);
                this->pictureBox1->SizeMode =
System::Windows::Forms::PictureBoxSi
zeMode::StretchImage;
                this->pictureBox1->TabIndex = 18;
```

```
                this->pictureBox1->TabStop = false;
                //
                // FormRegistro
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->BackColor = System::Drawing::Color::DarkSlateGray;
                this->ClientSize = System::Drawing::Size(774, 560);
                this->Controls->Add(this->pictureBox1);
                this->Controls->Add(this->dataGridView1);
                this->Controls->Add(this->label8);
                this->Controls->Add(this->button1);
                this->Controls->Add(this->textBox7);
                this->Controls->Add(this->label7);
                this->Controls->Add(this->textBox6);
                this->Controls->Add(this->label6);
                this->Controls->Add(this->textBox5);
                this->Controls->Add(this->label5);
                this->Controls->Add(this->textBox4);
                this->Controls->Add(this->label4);
                this->Controls->Add(this->textBox3);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->textBox2);
                this->Controls->Add(this->label2);
                this->Controls->Add(this->textBox1);
                this->Controls->Add(this->label1);
                this->Name = L"FormRegistro";
                this->Text = L"FormRegistro";
                this->Load += gcnew System::EventHandler(this, &FormRegistro::FormRegistro_Load);

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();
                this->ResumeLayout(false);
                this->PerformLayout();

        }
#pragma endregion
        private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {
        }
        private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
                String^ nom = textBox1->Text;
                String^ tip = textBox2->Text;
                String^ mar = textBox3->Text;
                String^ ubi = textBox4->Text;
                String^ est = textBox5->Text;

                if (textBox6->Text != "" && textBox7->Text != "") {
                        try {
                                double p_entrada = Convert::ToDouble(textBox6->Text);
                                double p_salida =
```

```
Convert::ToDouble(textBox7->Text);

                                    double
eficiencia = (p_salida / p_entrada) *
100;

                                    //
Guardamos en la BD

        ConexionBD::InsertarEquipo(no
m, tip, mar, ubi, est, p_entrada, p_salida,
eficiencia);


        MessageBox::Show("¡Equipo
registrado! Eficiencia calculada: " +
eficiencia + "%");

                                    // ---
CAMBIO 3: Refrescar la tabla al
guardar ---

        ActualizarTabla();

                                    //
Limpiar todo...

        textBox1->Clear(); textBox2-
>Clear(); textBox3->Clear();

        textBox4->Clear(); textBox5-
>Clear();

        textBox6->Clear(); textBox7-
>Clear();
                        }
                        catch (...) {

        MessageBox::Show("Error: Por
favor ingresa solo números en las
potencias.");
                        }
                }
                else {

        MessageBox::Show("Por favor
llena todos los datos, incluyendo
potencias.");
                }
        }
        private: System::Void
textBox2_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
textBox3_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
textBox4_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
textBox5_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
textBox6_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
textBox7_TextChanged(System::Object
^ sender, System::EventArgs^ e) {
        }
        private: System::Void
label1_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
label2_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
label3_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
label4_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
FormRegistro_Load(System::Object^
sender, System::EventArgs^ e) {
        }
        private: System::Void
label5_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
label6_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
```

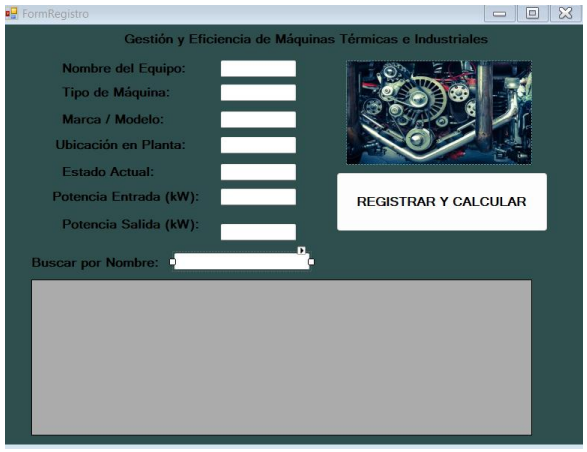
```
        private: System::Void
label7_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        private: System::Void
label8_Click(System::Object^ sender,
System::EventArgs^ e) {
        }
        };
}
```

4. Al poner todos los datos quedaría de la siguiente manera se podrá ver la tabla.

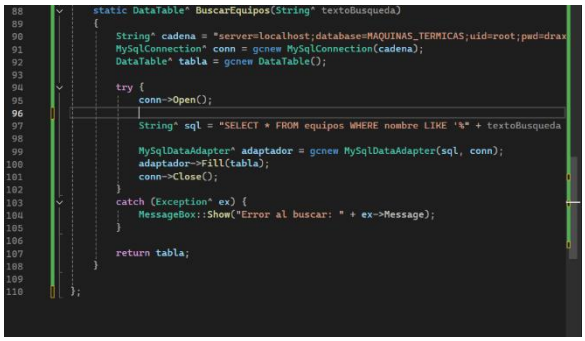


PROBLEMA 4
BÚSQUEDA DINÁMICA POR NOMBRE

1. Primero agregamos un label y texbox ubicada en el cuadro de herramientas para poder usar el buscador.



2. En el formulario Pillpe09a.h para poder usar el buscador con la opción de textbox agregamos el siguiente código en la parte final.

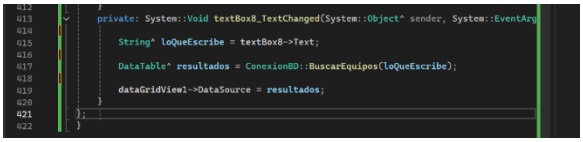


```
static DataTable^ BuscarEquipos(String^ textoBusqueda)
{
    String^ cadena = "server=localhost;database=MAQUINAS_TERMICAS;uid=root;pwd=drax
    MySqlConnection^ conn = gcnew MySqlConnection(cadena);
    DataTable^ tabla = gcnew DataTable();

    try {
        conn->Open();

        String^ sql = "SELECT * FROM equipos WHERE nombre LIKE '%" + textoBusqueda

        MySqlDataAdapter^ adaptador = gcnew MySqlDataAdapter(sql, conn);
        adaptador->Fill(tabla);
        conn->Close();
    }
    catch (Exception^ ex) {
        MessageBox::Show("Error al buscar: " + ex->Message);
    }

    return tabla;
}
};
```

```
#pragma once

using namespace System;
using namespace
System::Windows::Forms;
using namespace
MySql::Data::MySqlClient;
using namespace System::Data; //
IMPORTANTE PARA LA TABLA

public ref class ConexionBD
{
public:
    static bool ValidarAcceso(String^
user, String^ pass)
    {
        String^ cadenaConexion =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadenaConexion);

        try
        {
            conn->Open();
            String^ sql = "SELECT *
FROM usuarios WHERE usuario = '" +
user + "' AND password = '" + pass +
"'";
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            MySqlDataReader^ lector =
cmd->ExecuteReader();

            if (lector->Read())
            {
                conn->Close();
                return true;
            }
            else
            {
                conn->Close();
                return false;
            }
        }
        catch (Exception^ ex)
        {
            MessageBox::Show("Error de
conexión: " + ex->Message);
            return false;
        }
    }

    static void InsertarEquipo(String^
nom, String^ tip, String^ mar, String^
ubi, String^ est, double p_ent, double
p_sal, double efic)
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);

        try {
            conn->Open();
```

```
            String^ sql = "INSERT INTO
equipos (nombre, tipo, marca,
ubicacion, estado, potencia_entrada,
potencia_salida, eficiencia) VALUES
('" + nom + "', '" + tip + "', '" +
mar + "', '" + ubi + "', '" + est +
"', " + p_ent + ", " + p_sal + ", " +
efic + ")";
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            cmd->ExecuteNonQuery();
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
guardar en BD: " + ex->Message);
        }
    }


    static DataTable^ ObtenerEquipos()
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);
        DataTable^ tabla = gcnew
DataTable();

        try {
            conn->Open();
            String^ sql = "SELECT *
FROM equipos";
            MySqlDataAdapter^
adaptador = gcnew
MySqlDataAdapter(sql, conn);
            adaptador->Fill(tabla);
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
cargar lista: " + ex->Message);
        }

        return tabla;
    }


    static DataTable^
BuscarEquipos(String^ textoBusqueda)
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);
        DataTable^ tabla = gcnew
DataTable();

        try {
            conn->Open();

            String^ sql = "SELECT *
FROM equipos WHERE nombre LIKE '%" +
textoBusqueda + "%'";

            MySqlDataAdapter^
adaptador = gcnew
MySqlDataAdapter(sql, conn);
            adaptador->Fill(tabla);
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
buscar: " + ex->Message);
        }

        return tabla;
    }

};
```

3. Para que funcione el textbox agregamos el siguiente el código en el formulario FormRegistro primero damos doble click al textbox y aparezca la opción y debajo pegamos es código.



private: System::Void textBox8_TextChanged(System::Object^ sender, System::EventArgs^ e) {

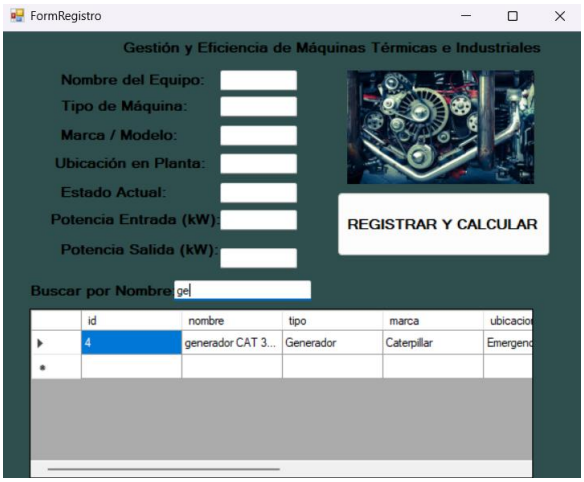
String^ loQueEscribe = textBox8->Text;

DataTable^ resultados = ConexionBD::BuscarEquipos(loQueEscribe);

dataGridView1->DataSource = resultados;

}

4. Al ir al buscador quedaría de la siguiente manera.

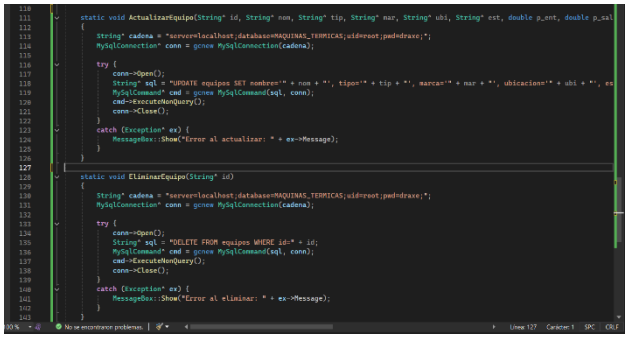


ACTUALIZAR y ELIMINAR DATOS DE DATOS MySQL

PILLPE ROMANI FRANK

1. Continuando los pasos de la clase de la semana 09, esta guía de la semana 10 sería la continuación:
   Parte 01:
   https://drive.google.com/file/d/1Y7El_eUBRwfOEr7KmFf7b4jBHyvr8vWp/view
   parte 02:
   https://drive.google.com/file/d/1mOpVfD1zhsyrk-t9rWa3Ao34Pj9v9_F3/view
   parte 03:
   https://drive.google.com/file/d/19B9d3fsN6ez1CSi1p_F-I6mpaY4B_iNQ/view
   parte04:
   https://drive.google.com/file/d/1pFaJkKtmvhf9eNkI0wMylVk-8Er8a3VG/view

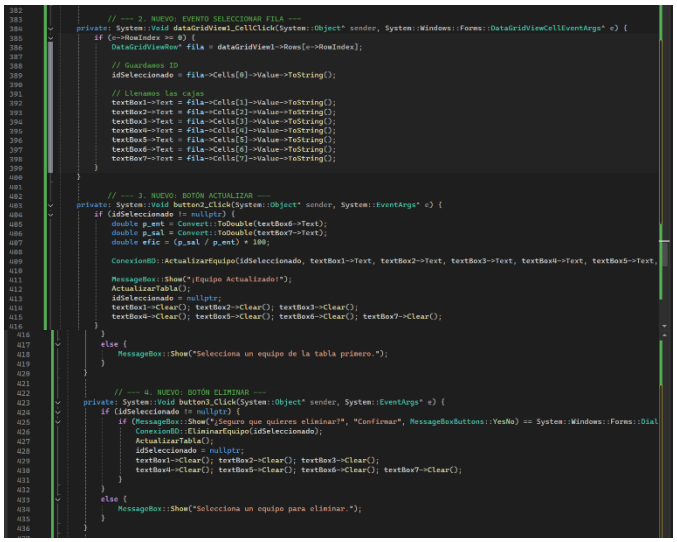
2. Se añadieron dos métodos estáticos a la clase de conexión para manejar las sentencias SQL de manipulación esto es en el formulario pillpe09a.h.
   Código 1: Métodos de Actualizar y Eliminar en C++ / MySQL.



```
    static void
ActualizarEquipo(String^ id, String^
nom, String^ tip, String^ mar, String^
ubi, String^ est, double p_ent, double
p_sal, double efic)
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);

        try {
            conn->Open();
            String^ sql = "UPDATE
equipos SET nombre='" + nom + "',
tipo='" + tip + "', marca='" + mar +
"', ubicacion='" + ubi + "', estado='"
+ est + "', potencia_entrada=" + p_ent
+ ", potencia_salida=" + p_sal + ",
eficiencia=" + efic + " WHERE id=" +
id;
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            cmd->ExecuteNonQuery();
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
actualizar: " + ex->Message);
        }
    }

    static void EliminarEquipo(String^
id)
    {
        String^ cadena =
"server=localhost;database=MAQUINAS_TE
RMICAS;uid=root;pwd=draxe;";
        MySqlConnection^ conn = gcnew
MySqlConnection(cadena);

        try {
            conn->Open();
            String^ sql = "DELETE FROM
equipos WHERE id=" + id;
            MySqlCommand^ cmd = gcnew
MySqlCommand(sql, conn);
            cmd->ExecuteNonQuery();
            conn->Close();
        }
        catch (Exception^ ex) {
            MessageBox::Show("Error al
eliminar: " + ex->Message);
        }
    }

      };
```

3. Lógica de Interfaz de Usuario Se programó la interacción visual para que los botones respondan a la selección del usuario agregando 2 button y modificando el código de dataGridView en FormRegistro.h.

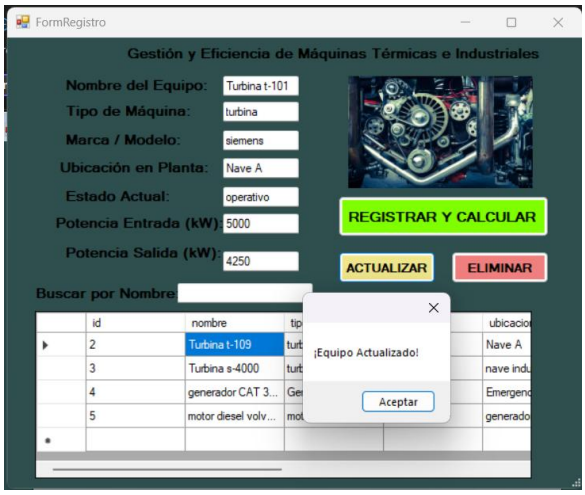
Código 2: Eventos de Selección y Botones de Acción.

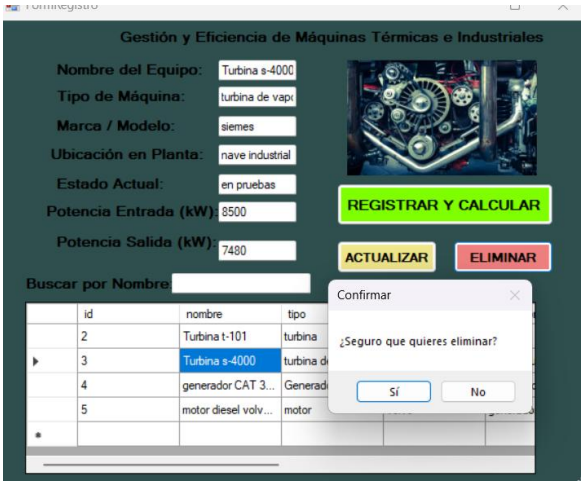


```
        // --- 2. NUEVO: EVENTO SELECCIONAR FILA ---
private: System::Void dataGridView1_CellClick(System::Object^ sender,
System::Windows::Forms::DataGridViewCellEventArgs^ e) {
    if (e->RowIndex >= 0) {
        DataGridViewRow^ fila = dataGridView1->Rows[e->RowIndex];

        // Guardamos ID
        idSeleccionado = fila->Cells[0]->Value->ToString();

        // Llenamos las cajas
        textBox1->Text = fila->Cells[1]->Value->ToString();
        textBox2->Text = fila->Cells[2]->Value->ToString();
        textBox3->Text = fila->Cells[3]->Value->ToString();
        textBox4->Text = fila->Cells[4]->Value->ToString();
        textBox5->Text = fila->Cells[5]->Value->ToString();
        textBox6->Text = fila->Cells[6]->Value->ToString();
        textBox7->Text = fila->Cells[7]->Value->ToString();
    }
}

        // --- 3. NUEVO: BOTÓN ACTUALIZAR ---
private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
    if (idSeleccionado != nullptr) {
        double p_ent = Convert::ToDouble(textBox6->Text);
        double p_sal = Convert::ToDouble(textBox7->Text);
        double efic = (p_sal / p_ent) * 100;


    ConexionBD::ActualizarEquipo(idSeleccionado, textBox1->Text, textBox2->Text, textBox3->Text, textBox4->Text, textBox5->Text, p_ent, p_sal, efic);

        MessageBox::Show("¡Equipo Actualizado!");
        ActualizarTabla();
        idSeleccionado = nullptr;
        textBox1->Clear(); textBox2->Clear(); textBox3->Clear();
        textBox4->Clear(); textBox5->Clear(); textBox6->Clear(); textBox7->Clear();
    }
    else {

    MessageBox::Show("Selecciona un equipo de la tabla primero.");
    }
}

        // --- 4. NUEVO: BOTÓN ELIMINAR ---
private: System::Void button3_Click(System::Object^ sender,
System::EventArgs^ e) {
    if (idSeleccionado != nullptr) {
        if (MessageBox::Show("¿Seguro que quieres eliminar?", "Confirmar", MessageBoxButtons::YesNo) == System::Windows::Forms::DialogResult::Yes) {

    ConexionBD::EliminarEquipo(idSeleccionado);
                ActualizarTabla();
                idSeleccionado = nullptr;
```

```
                    textBox1->Clear();
textBox2->Clear(); textBox3->Clear();
                    textBox4->Clear();
textBox5->Clear(); textBox6->Clear();
textBox7->Clear();
            }
        }
        else {

        MessageBox::Show("Selecciona un
equipo para eliminar.");
        }
        }
```

4. Usaremos la opción depurar de visual studio 2022 y quedaría de la siguiente manera.

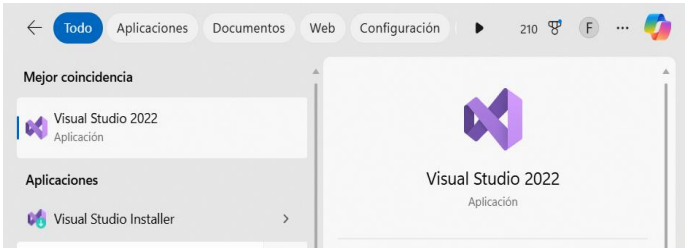




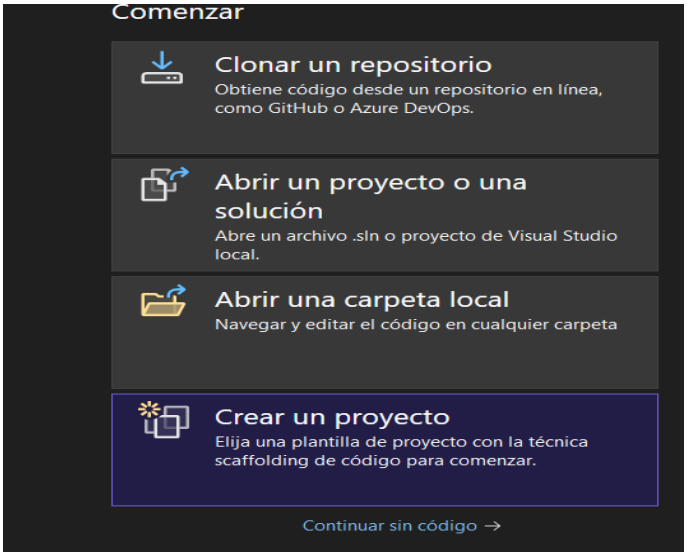
GUIA DE FUNCIONES GRÁFICAS

Pillpe Romaní

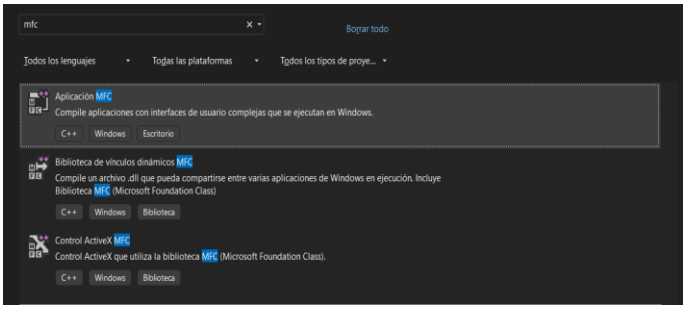
1) El primer paso sería ir al buscador y escribir visual studio 2022.



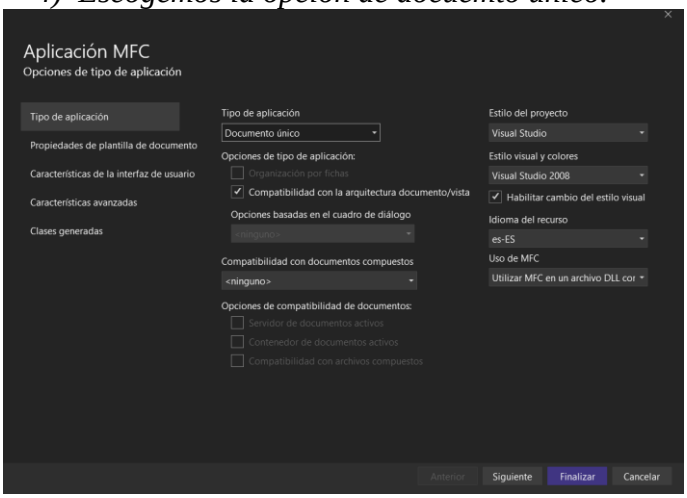
2) AL abrir el visual studio 2022 vamos a la opcion de crear un proyecto.



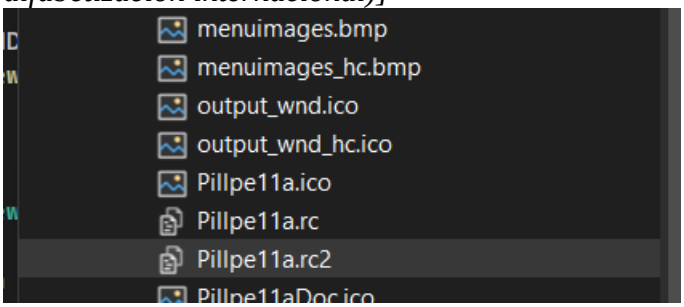
*3) Buscamos la opcion de **aplicación MFC***

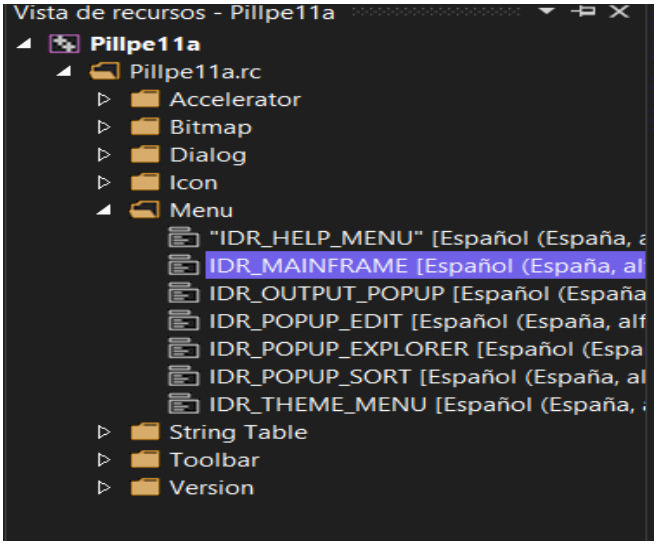


4) Escogemos la opcion de docuemto unico.

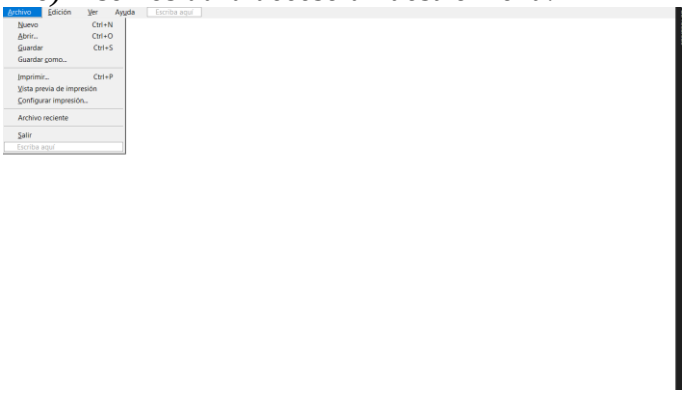


5) En el explorador de soluciones buscamos la opcion de Pillpe11a.rc2, despues de darle click buscamos la opcion de menu y damos doble click IDR_MAINFRAME [Español (España, alfabetización internacional)]

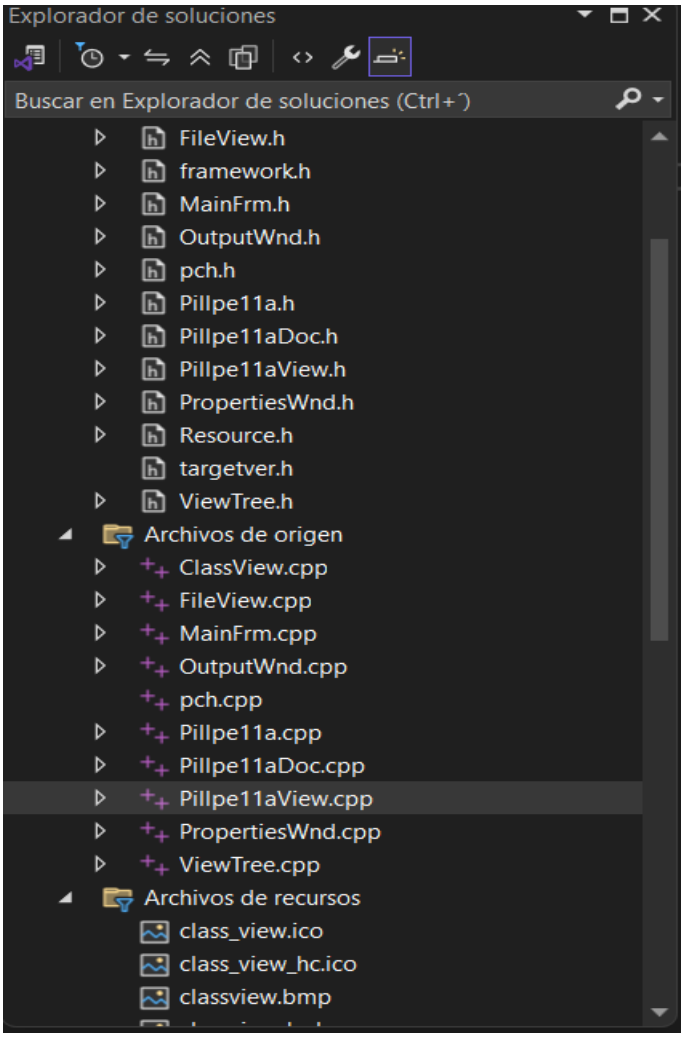


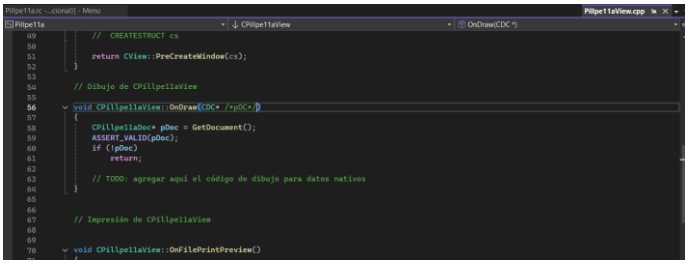


6) *Eso nos dará acceso a nuestro menú.*



7) *El siguiente paso es ir al explorador de soluciones y buscar la opción de* ***Pillpe11a.view.cpp*** *desplegando nuestro código.*





8) *Codigo para pillpe11a.view.cpp*

```
// Pillpe11aView.cpp: implementación de la
clase CPillpe11aView
//

#include "pch.h"      // ESTO DEBE IR PRIMERO
SIEMPRE
#include "framework.h"
#include <math.h>     // <--- NECESARIO PARA
LA FUNCIÓN SENO

#ifndef SHARED_HANDLERS
#include "Pillpe11a.h"
#endif

#include "Pillpe11aDoc.h"
#include "Pillpe11aView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CPillpe11aView

IMPLEMENT_DYNCREATE(CPillpe11aView, CView)

BEGIN_MESSAGE_MAP(CPillpe11aView, CView)
  // He quitado la línea del menú que te daba
error (ID_GRAFICO_SENO)
  ON_COMMAND(ID_FILE_PRINT,
&CView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_DIRECT,
&CView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_PREVIEW,
&CPillpe11aView::OnFilePrintPreview)
  ON_WM_CONTEXTMENU()
  ON_WM_RBUTTONUP()
END_MESSAGE_MAP()

// Construcción o destrucción de
CPillpe11aView

CPillpe11aView::CPillpe11aView() noexcept
{
}

CPillpe11aView::~CPillpe11aView()
{
}

BOOL
CPillpe11aView::PreCreateWindow(CREATESTRUCT&
cs)
```

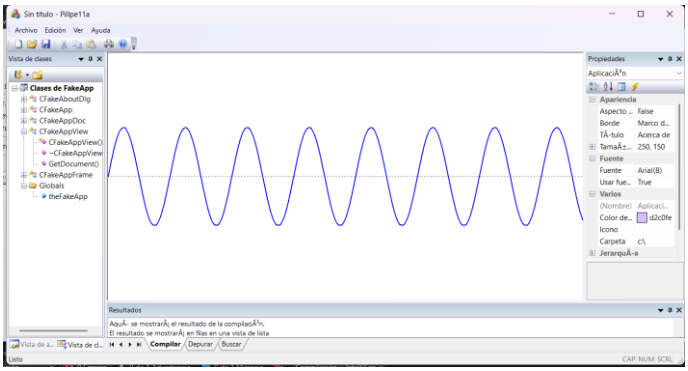
```
{
  return CView::PreCreateWindow(cs);
}

// ----------------------------------------
---------------------
// AQUÍ ESTÁ EL CÓDIGO DEL GRÁFICO (Se
ejecuta automático)
// ----------------------------------------
---------------------
void CPillpe11aView::OnDraw(CDC* pDC)
{
  CPillpe11aDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  if (!pDoc)
    return;

  // 1. Obtener dimensiones de la pantalla
  CRect rect;
  GetClientRect(&rect);

  int ejeY = rect.Height() / 2; // Mitad de
la pantalla (Alto)
  int ancho = rect.Width();     // Ancho
total

  // 2. DIBUJAR EL EJE X (Línea gris
punteada)
  CPen lapizEje(PS_DOT, 1, RGB(100, 100,
100));
  CPen* pOldPen = pDC-
>SelectObject(&lapizEje);

  pDC->MoveTo(0, ejeY);
  pDC->LineTo(ancho, ejeY);

  // 3. DIBUJAR LA ONDA SENOIDAL (Línea Azul
Gruesa)
  CPen lapizSeno(PS_SOLID, 2, RGB(0, 0,
255)); // Azul, grosor 2
  pDC->SelectObject(&lapizSeno);

  // Nos ubicamos al inicio (Izquierda)
  pDC->MoveTo(0, ejeY);

  // Recorremos todo el ancho de la pantalla
pixel por pixel
  for (int x = 0; x < ancho; x++)
  {
    // FÓRMULA MATEMÁTICA:
    // y = Centro - (Amplitud * seno(x *
Frecuencia))

    double valorSeno = sin(x * 0.05); // 0.05
define qué tan "juntas" están las ondas
    int y = ejeY - (int)(100 * valorSeno); //
100 es la altura de la onda

    pDC->LineTo(x, y);
  }

  // Restaurar el pincel original para no
dejar basura en memoria
  pDC->SelectObject(pOldPen);
}


// --- Funciones del Sistema (No tocar) ---

void CPillpe11aView::OnFilePrintPreview()
{
#ifndef SHARED_HANDLERS
  AFXPrintPreview(this);
#endif
}

BOOL
CPillpe11aView::OnPreparePrinting(CPrintInfo*
pInfo)
{
  return DoPreparePrinting(pInfo);
}

void CPillpe11aView::OnBeginPrinting(CDC*
/*pDC*/, CPrintInfo* /*pInfo*/)
{
}

void CPillpe11aView::OnEndPrinting(CDC*
/*pDC*/, CPrintInfo* /*pInfo*/)
{
}

void CPillpe11aView::OnRButtonUp(UINT /*
nFlags */, CPoint point)
{
  ClientToScreen(&point);
  OnContextMenu(this, point);
}

void CPillpe11aView::OnContextMenu(CWnd* /*
pWnd */, CPoint point)
{
#ifndef SHARED_HANDLERS
  theApp.GetContextMenuManager()-
>ShowPopupMenu(IDR_POPUP_EDIT, point.x,
point.y, this, TRUE);
#endif
}

#ifdef _DEBUG
void CPillpe11aView::AssertValid() const
{
  CView::AssertValid();
}

void CPillpe11aView::Dump(CDumpContext& dc)
const
{
  CView::Dump(dc);
}

CPillpe11aDoc* CPillpe11aView::GetDocument()
const
{
```

```
  ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CPillpe11aDoc)));
  return (CPillpe11aDoc*)m_pDocument;
}
#endif
```

9) Código para pillpe11aView.h

```
// Pillpe11aView.h: interfaz de la clase
CPillpe11aView
//

#pragma once

class CPillpe11aView : public CView
{
protected: // Crear sólo a partir de
serialización
  CPillpe11aView() noexcept;
  DECLARE_DYNCREATE(CPillpe11aView)

  // Atributos
public:
  CPillpe11aDoc* GetDocument() const;

  // Operaciones
public:

  // Reemplazos
public:
  virtual void OnDraw(CDC* pDC);  //
Reemplazado para dibujar esta vista
  virtual BOOL PreCreateWindow(CREATESTRUCT&
cs);
protected:
  virtual BOOL OnPreparePrinting(CPrintInfo*
pInfo);
  virtual void OnBeginPrinting(CDC* pDC,
CPrintInfo* pInfo);
  virtual void OnEndPrinting(CDC* pDC,
CPrintInfo* pInfo);

  // Implementación
public:
  virtual ~CPillpe11aView();
#ifdef _DEBUG
  virtual void AssertValid() const;
  virtual void Dump(CDumpContext& dc) const;
#endif

protected:

  // Funciones de asignación de mensajes
generadas
protected:
  afx_msg void OnFilePrintPreview();
  afx_msg void OnRButtonUp(UINT nFlags,
CPoint point);
  afx_msg void OnContextMenu(CWnd* pWnd,
CPoint point);

  // --- ESTA ES LA LÍNEA QUE TE FALTABA ---
  afx_msg void OnGraficoSeno();
  // ---------------------------------------
  DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // Versión de depuración en
Pillpe11aView.cpp
inline CPillpe11aDoc*
CPillpe11aView::GetDocument() const
{
  return
reinterpret_cast<CPillpe11aDoc*>(m_pDocument)
;
}
#endif
```

10) Ejecutamos después de agregar los códigos y se vera de esta manera.

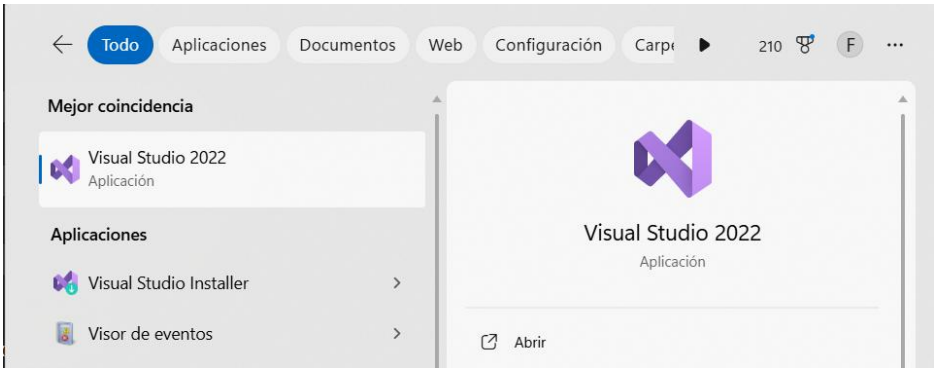


EVAP12

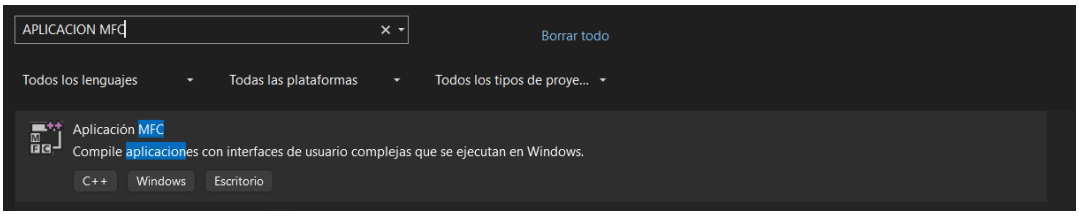
PROYECTO DE MENPU DE CONTROL INTERACTIVO

MENÚ DESPLEGABLE CON LA BIBLIOTECA MFC - MICROSOFT FOUNDATION CLASS DE VISUAL STUDIO C++ 2022 COMMUNITY

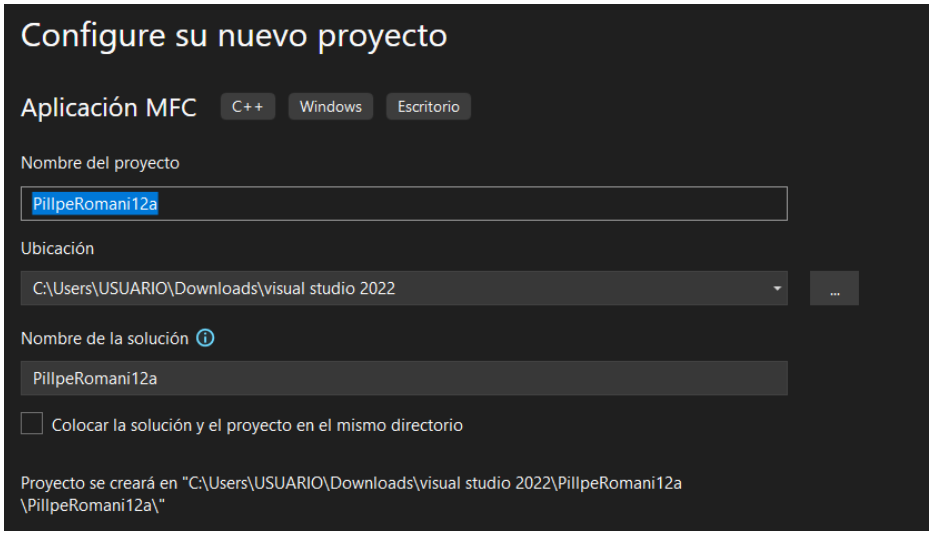
1. **Buscamos en el buscador la aplicación de visual studio 2022**



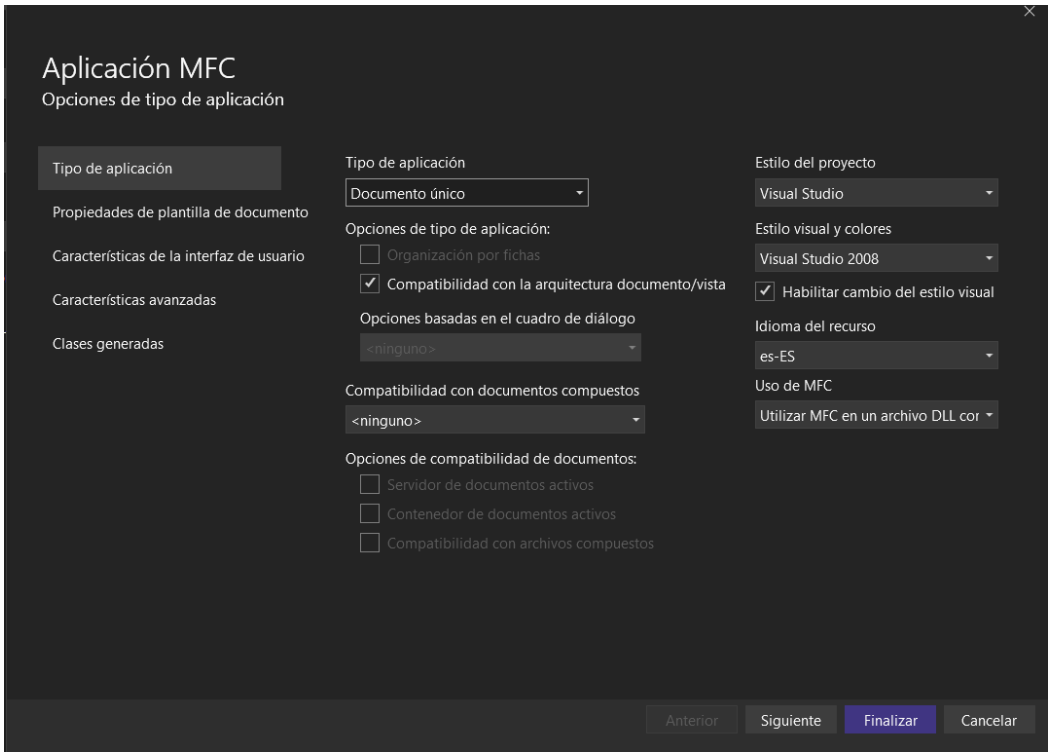
2. **En el buscador buscamos la opción de APLICACIÓN MFC.**



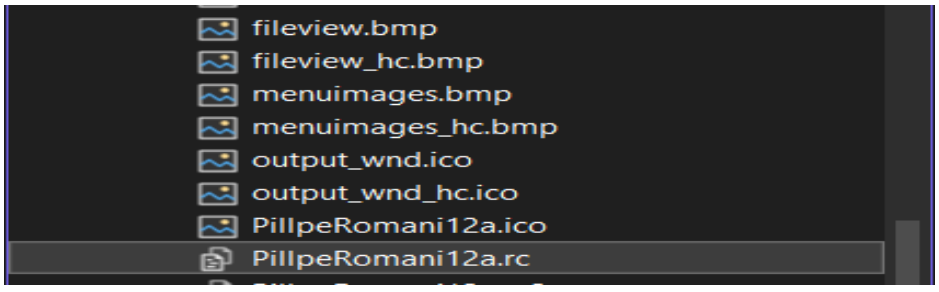
3. **Creamos un nuevo proyecto donde en el nombre de nuestro nuevo proyecto ponemos nuestro apellido, numero de clase, el intento.**



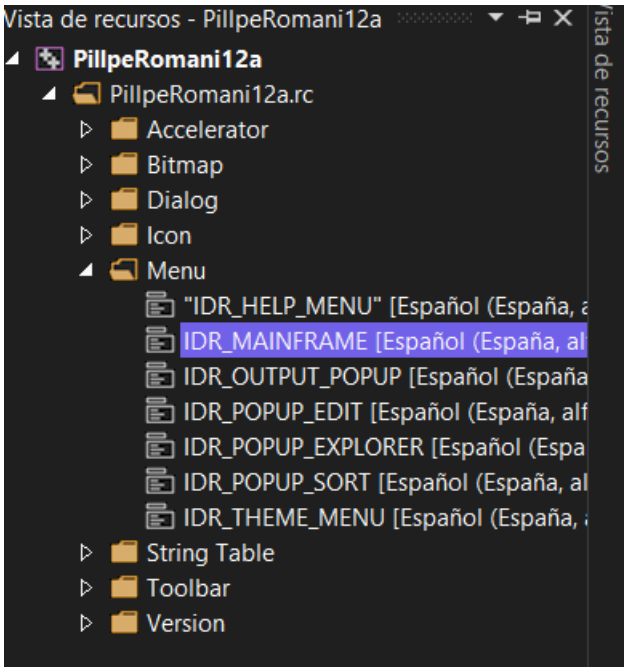
4. **En la configuración de aplicación MFC en la opción de tipo de aplicación escogemos la opción de documento único**



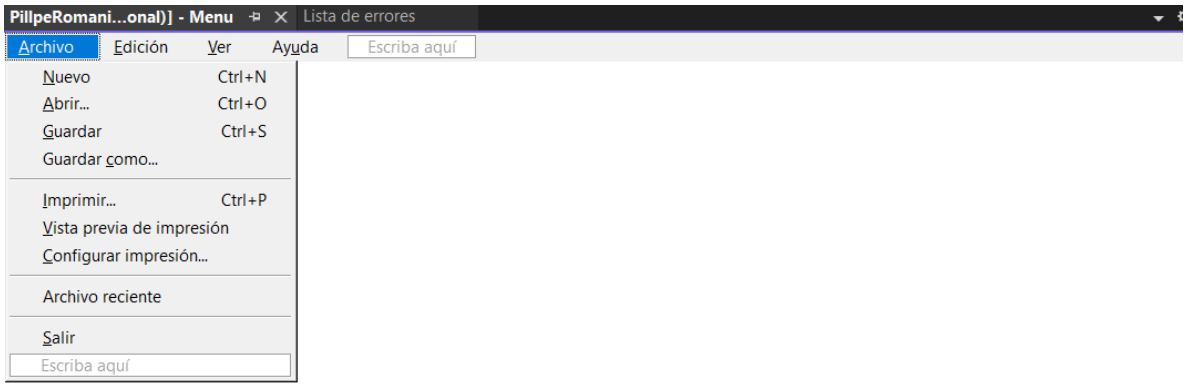
5. **En el explorador de soluciones buscamos las opciones del nombre de nuestro archivo creado con el final tenga .rc (PillpeRomani12a.rc)**



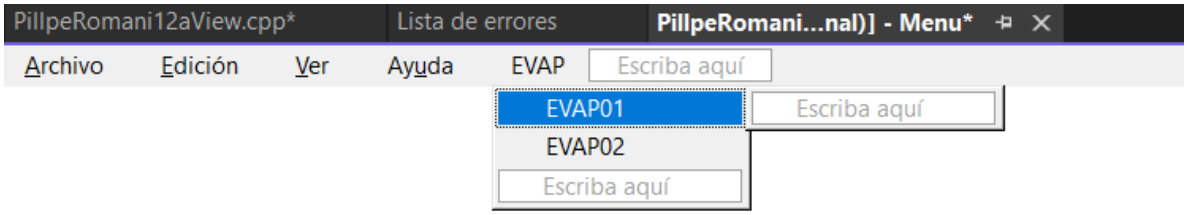
6. **Se nos Abrera las siguientes opciones donde buscamos la carpeta de MENU donde daremos doble clic en la segunda opción IDR_MAINFRAME [español (España, alfabetización internacional)]**



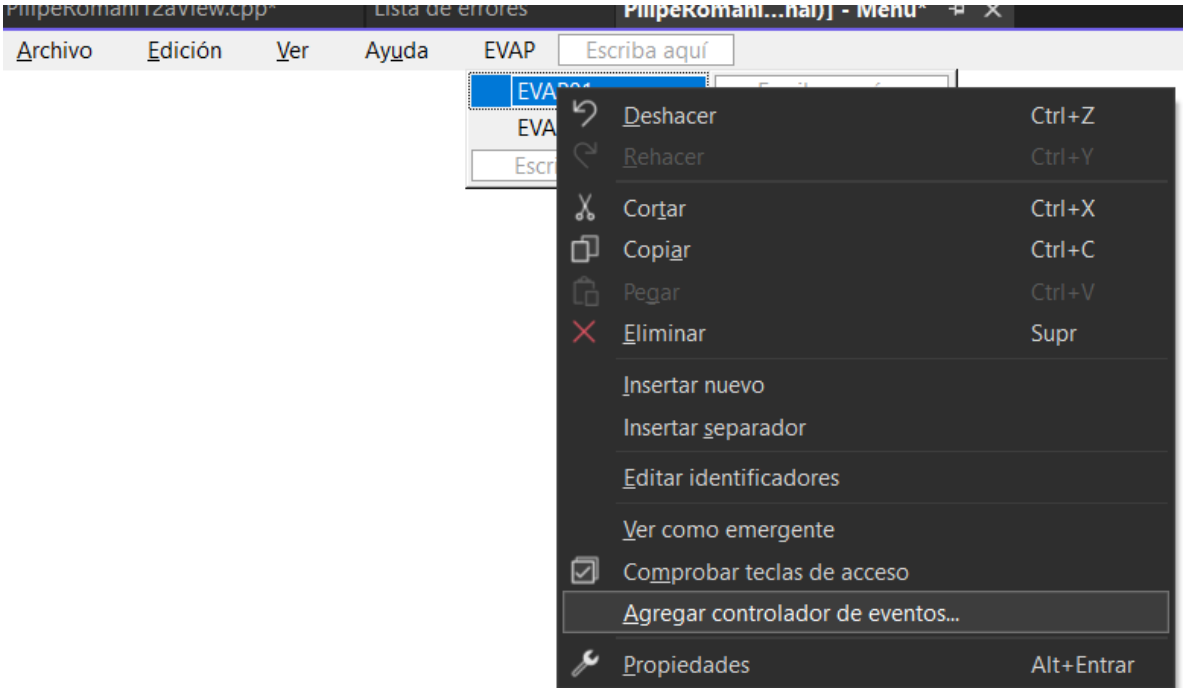
7. **Se abrirá el menú en la pantalla de inicio.**



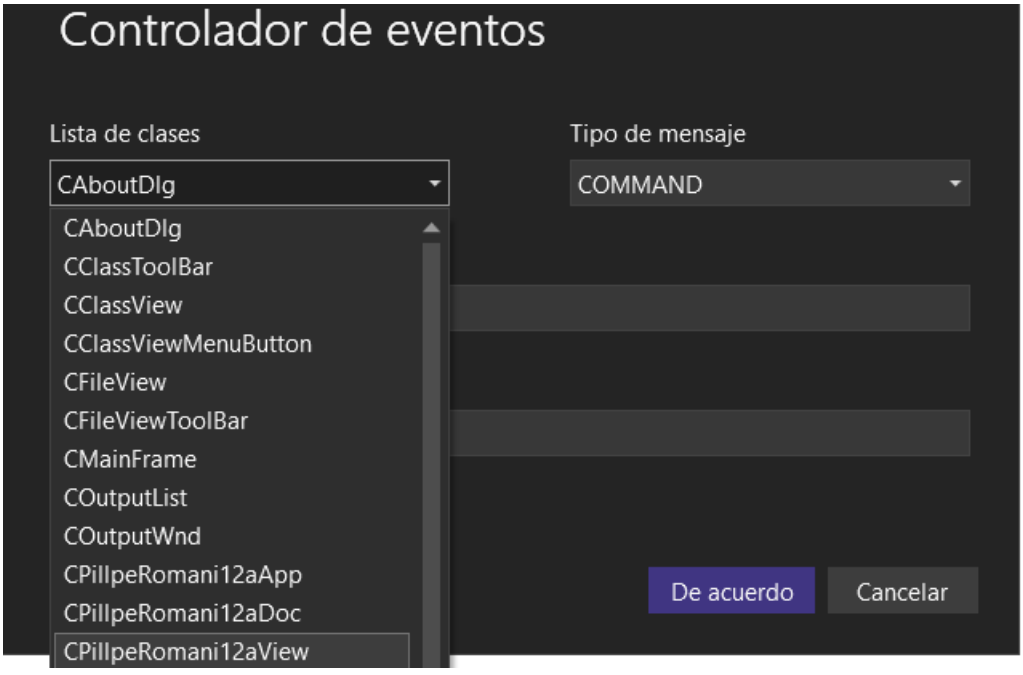
8. **En la parte superior a la derecha nos dice que escriba aquí, que es donde agregaremos nuestros proyectos.**



9. **Ahora buscaremos nuestros proyectos damos un clic derecho y optamos por la opción de agregar controlador de eventos.**



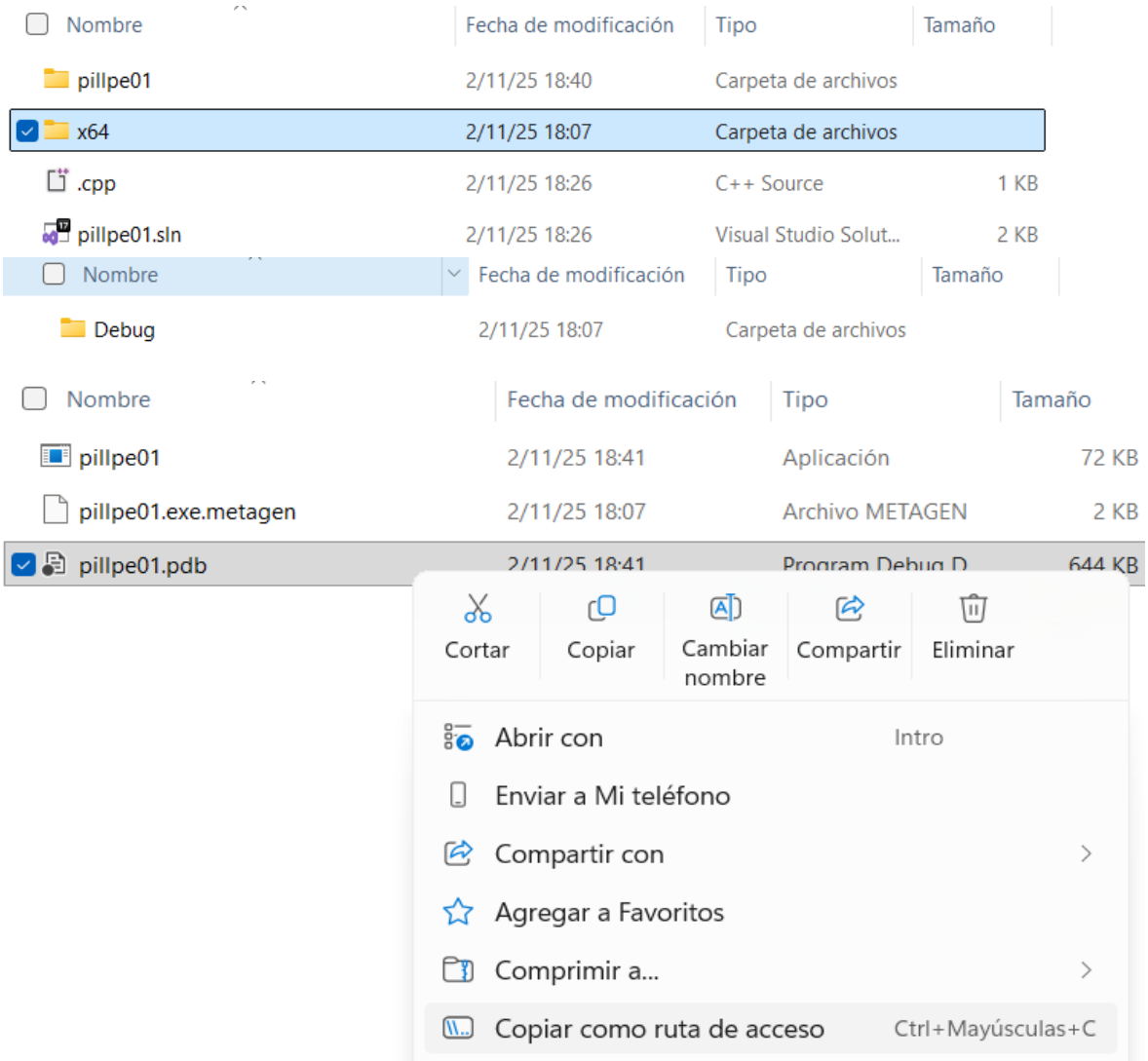
10. **EN EL CONTROLADOR DE EVENTOS solo cambiamos la opción de lista por el nombre de nuestro archivo que termine en view.**



11. **Se abrirá el código del menú donde agregaremos el siguiente código ubicada en la pag 95 del drive: https://drive.google.com/file/d/1erPpYFyTfBomSEFU-uqvTK3fQrD3p3zt/view**
Abrir un archivo ejecutable y externo a la aplicación del Menú:
ShellExecute(NULL, TEXT("open"),
TEXT("D:\\IIME\\EJECUTABLES\\FUNCION1.exe"), NULL, NULL,
SW_SHOWNORMAL);

```
128
129     // Controladores de mensajes de CPillpeRomani12aView
130
131     void CPillpeRomani12aView::OnEvapEvap01()
132     {
133
134     }
```

12. **Buscamos la carpeta que vamos a presentar, buscamos la carpeta x64, la carpeta debug y copiamos la ruta de acceso**



13. **Sobrescribimos sobre el código del drive copiado y agregamos la ruta de acceso.**

```
130
131  void CPillpeRomani12aView::OnEvapEvap01()
132  {
133      ShellExecute(NULL, TEXT("open"),
134          TEXT("C:\\Users\\USUARIO\\source\\repos\\pillpe01\\pillpe01.sln"),
135          NULL, NULL, SW_SHOWNORMAL);
136  }
137
```

14. **El menú se vería de esta manera y dándole clic a uno de los subtemas de evap nos llevara directo al archivo escogido.**